Getting the Most Out of Cocoa Bindings

Session 136

These are confidential sessions—please refrain from filming or taking pictures.

Application Technologies

Getting the Most Out of Cocoa Bindings

Ron Lue-Sang
Engineer
Cocoa Data/Core Bindings

http://developer.apple.com/wwdc2005/

Introduction

- Working with controllers
  - NSArrayController
  - NSTreeController
- Tips and Tricks for advanced bindings
  - Master/detail interfaces
  - Selection control bindings
  - Target/argument bindings
- Bonus bindings topics
  - NSControllers and Core Data
  - Error presentation
  - Debugging

NSArrayController

http://developer.apple.com/wwdc2005/

NSArrayController

- There are three important NSArrayController accessors
  - `arrangedObjects` proxy
  - `selection` proxy
  - `selectedObjects` array
- IB Actions

NSArrayController

`arrangedObjects` proxy

- Looks a lot like a regular NSArray
- Provides a view of the underlying NSArray
  - Sorted based on sortDescriptors
  - Filtered based on filterPredicate

arrangedObjects `= [arrayController arrangedObjects]`

NSArrayController

`arrangedObjects` proxy

- Looks a lot like a regular NSArray
- Provides a view of the underlying NSArray
  - Sorted based on sortDescriptors
  - Filtered based on filterPredicate

NSArrayController

`arrangedObjects` proxy

- Live proxy array whose content changes
  - Sorting and filtering
  - Add/remove
  - Controller content/content binding changes
- Never mutate the proxy
  - Copies won’t change underneath you

NSArrayController

`arrangedObjects` proxy and Filtering (new in Tiger!)

- Filtering is based on NSPredicate (Foundation)
  - Defines a logical condition used to constrain a search
  - Lots of functionality and flexibility
- Either bind or set the filterPredicate
- Use NSSearchField predicate bindings to control filtering

NSArrayController

`selection` **proxy**

- Provides a single access point for performing KVC on all selected objects
  - setValue:forKey: / setValue:forKeyPath:
  - valueForKey: / valueForKeyPath:
- Fully KVO compliant

Selection `= [arrayController selection]`

NSArrayController

`selection` proxy

- Ordinarily you see this

NSArrayController

`selection` **proxy**

- But what happens when there isn’t exactly one selected item with one name

?

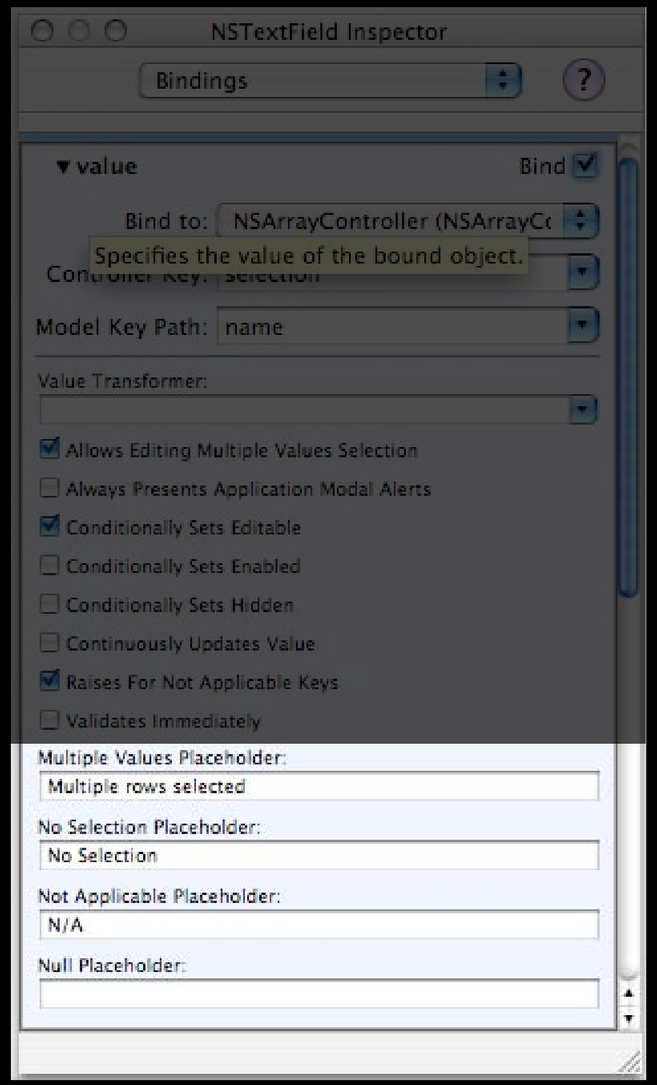
```
[selection valueForKeyPath:@”name”]
```

returns a marker

NSArrayController

`selection` proxy

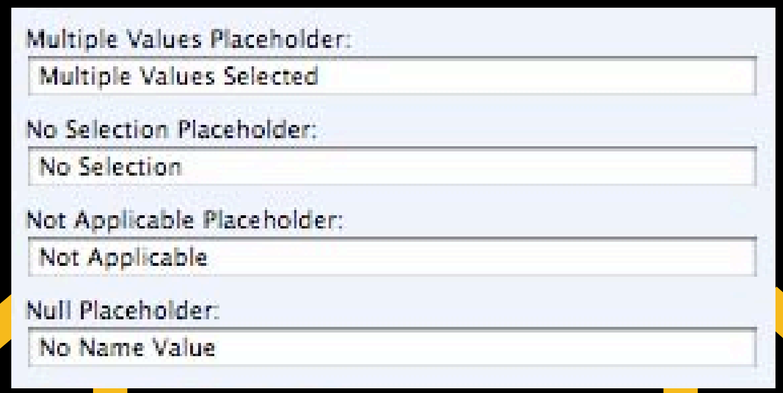
- Bound views automatically convert selection markers and nil into place holder values set on the binding

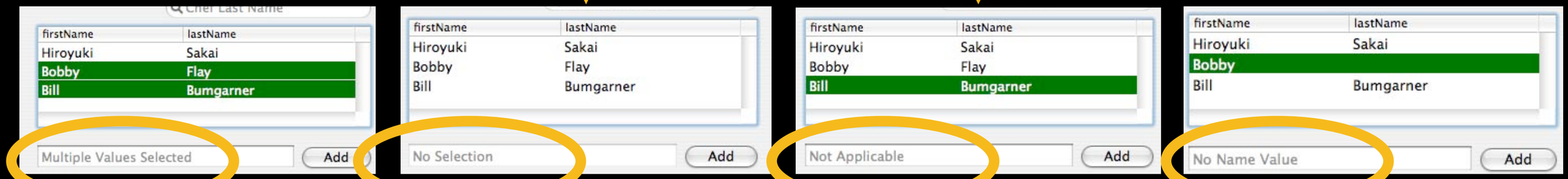


NSArrayController

`selection` proxy

- Bound views automatically convert selection markers and nil into place holder values set on the binding





NSArrayController

`selection` proxy

- Useful KVC operations for NSArray and Controller proxies

```
[ selection valueForKeyPath:@"@avg.salary"]
```

Constant	Value
`NSCountKeyValueOperator`	`@"count"`
`NSMaximumKeyValueOperator`	`@"max"`
`NSMinimumKeyValueOperator`	`@"min"`
`NSSumKeyValueOperator`	`@"sum"`

http://developer.apple.com/documentation/Cocoa/Conceptual/KeyValueCoding/Concepts/ArrayOperators.html

NSArrayController

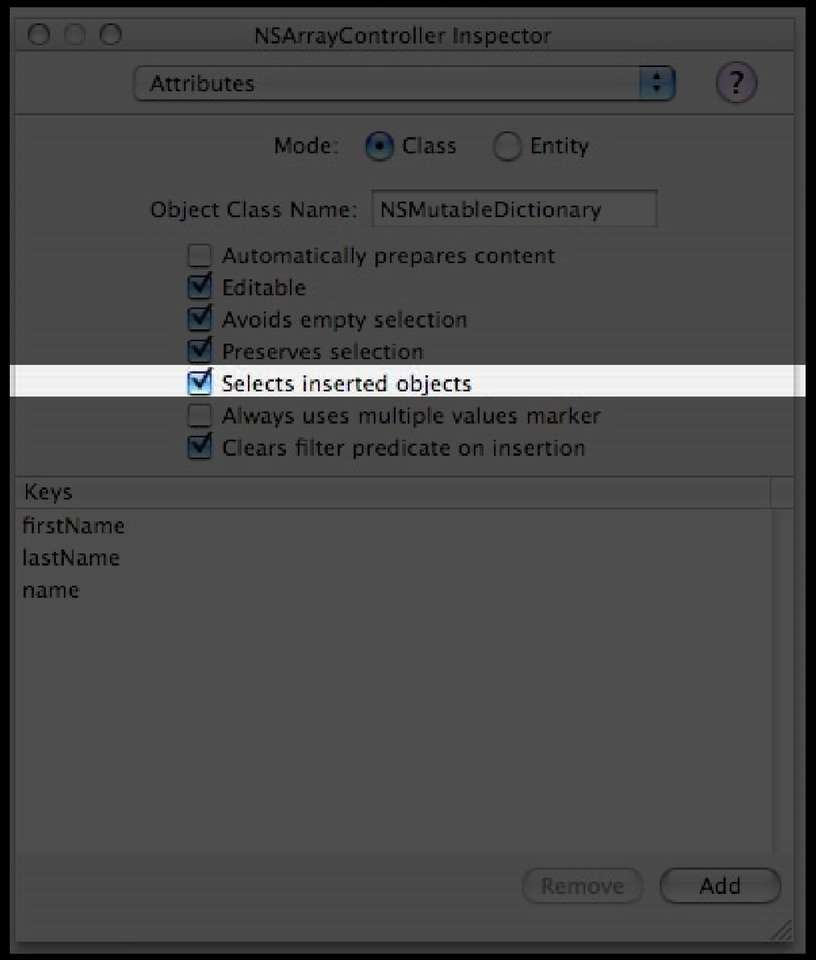
Influencing selection

- Four flags
  - Selects inserted objects
  - Avoids empty selection
  - Preserves selection
  - Always uses multiple selection marker

NSArrayController

Influencing selection

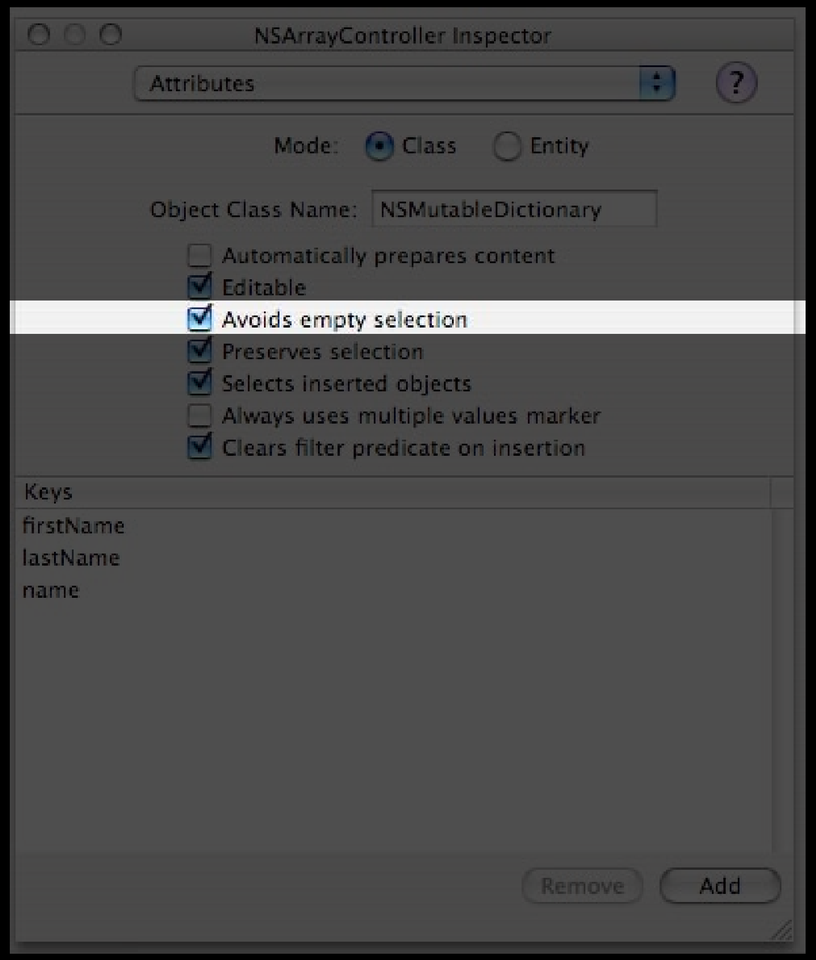
- Selects inserted objects

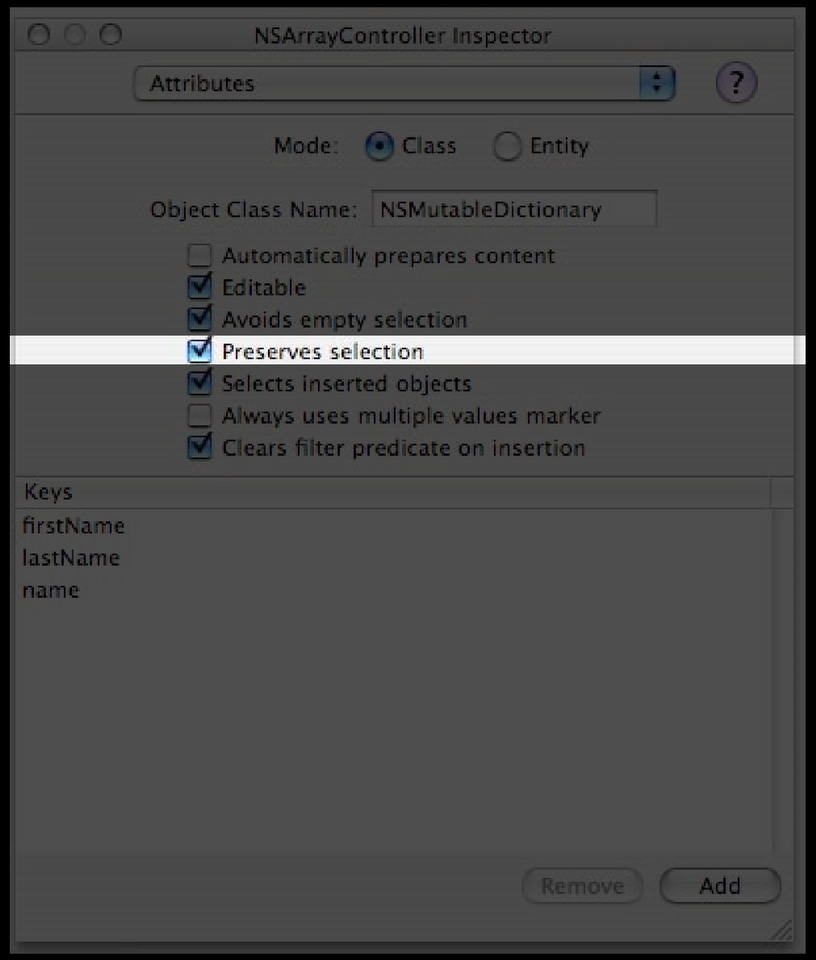


NSArrayController

Influencing selection

- Avoids empty selection





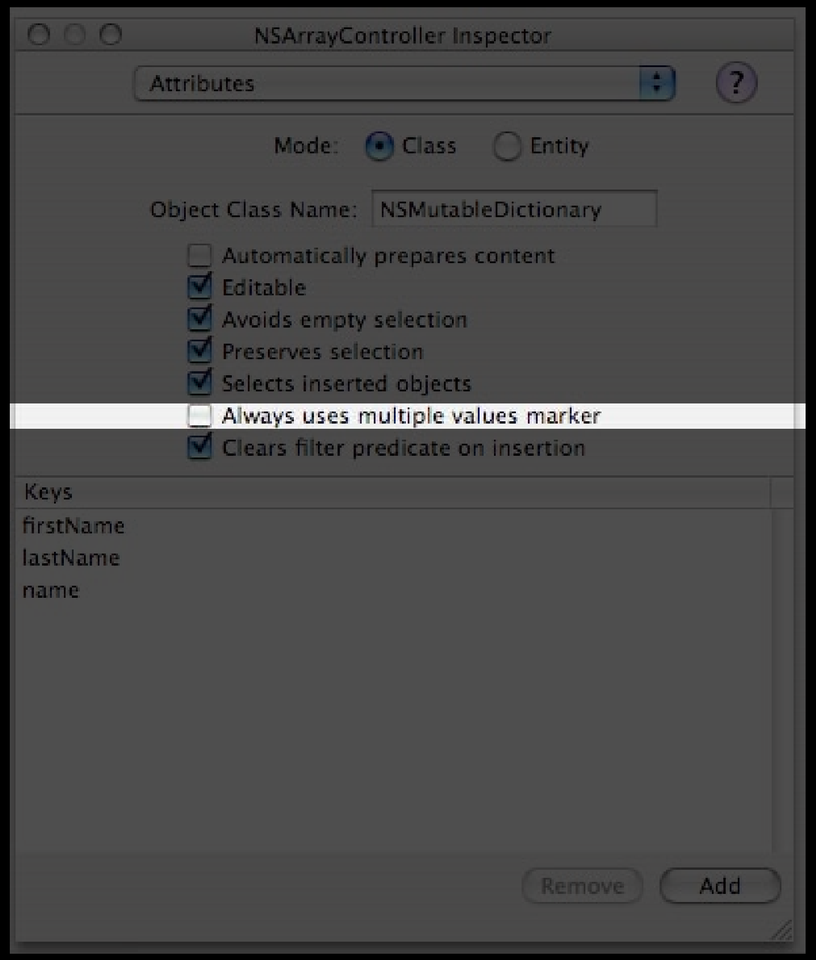
NSArrayController

Influencing selection

- Preserves selection
  - Has performance impact

NSArrayController

Influencing selection



- Always uses multiple values marker
  - Has performance impact

NSArrayController

`selectedObjects`

- NSArray of objects corresponding to the selection
  - Just an array (as opposed to a proxy)
  - Useful for handing to worker code or contentArrayBinding
    - Target/Argument bindings and Master/Detail Interfaces
- Avoid observing/binding through the array
  - More expensive than observing selection and arrangedObjects proxies
- Observing/Binding to selectedObjects is OK

`@"selectedObjects"`

`@"selectedObjects.name"`

`@"selection.name"`

NSArrayController

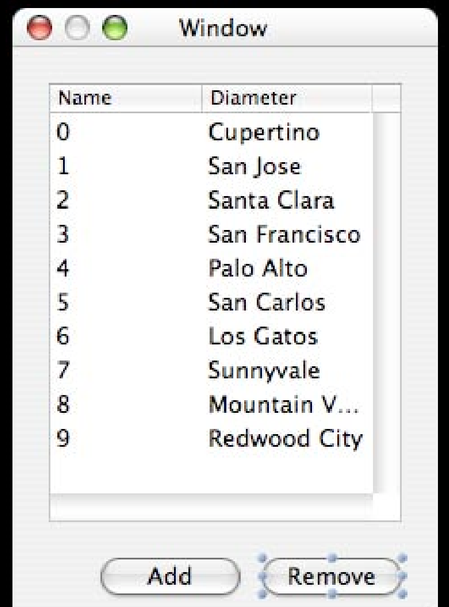
IB Actions

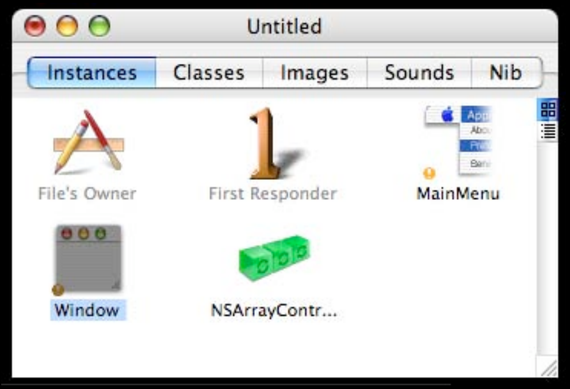
- NSArrayController API for mutating underlying content
  - add:(id)sender
  - insert:(id)sender
  - remove: (id)sender
- Bind to canAdd/canInsert/canRemove for enabling/disabling UI elements

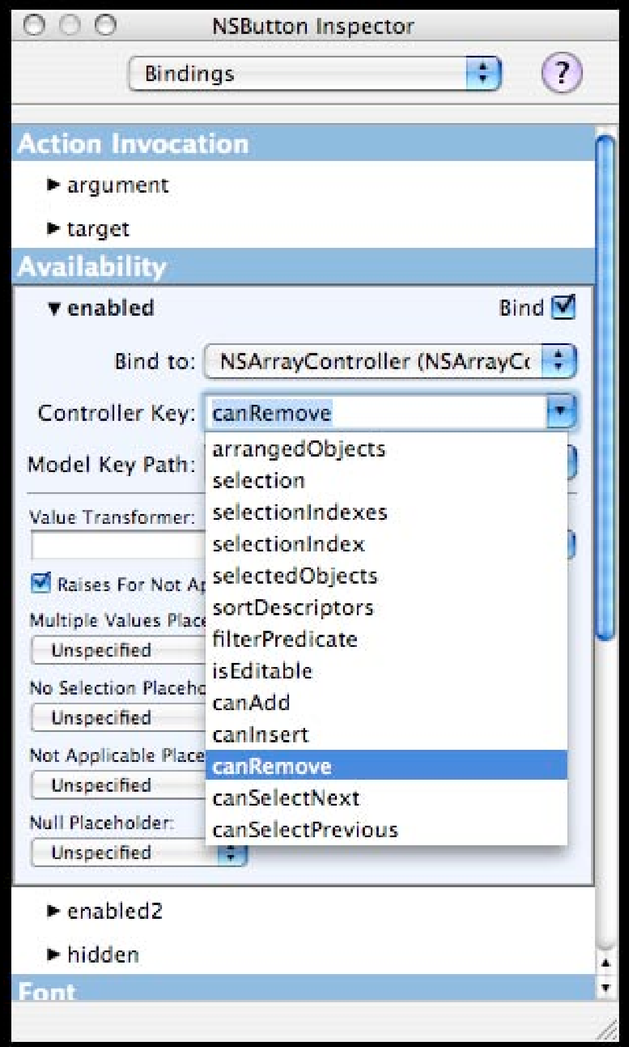
NSArrayController

IB Actions

- Enabling UI







NSArrayController

IB Actions

- Actions are performed delayed to support sheets for error reporting
- If you want immediate gratification and better performance…

NSArrayController

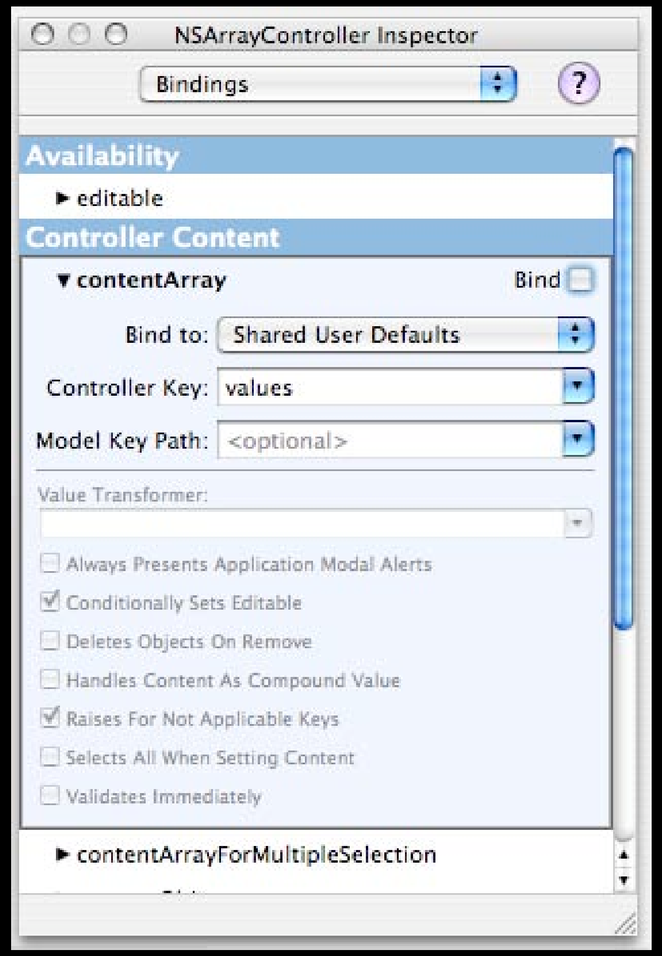
Content mutating methods

- NSArrayController API for mutating underlying content
  - `addObject(s):`
  - `insertObject(s):atArrangedObjectIndex(es)`
  - `removeObject(s)AtArrangedObjectIndex(es)`
- Controllers are optimized for these cases
  - Faster than modifying model objects directly

NSArrayController

Recap

- Never mutate the `arrangedObjects` proxy directly
- Never mutate the underlying array without sending a KVO notification
- Make sure model objects are fully KVO compliant



NSArrayController

Recap

- Avoid observing/binding through `selectedObjects`
- Avoid observing/binding through `content`
  - Different from binding content bindings of the controller

NSObjectController and NSTreeController

http://developer.apple.com/wwdc2005/

NS*Controller

Same Design/Behavior

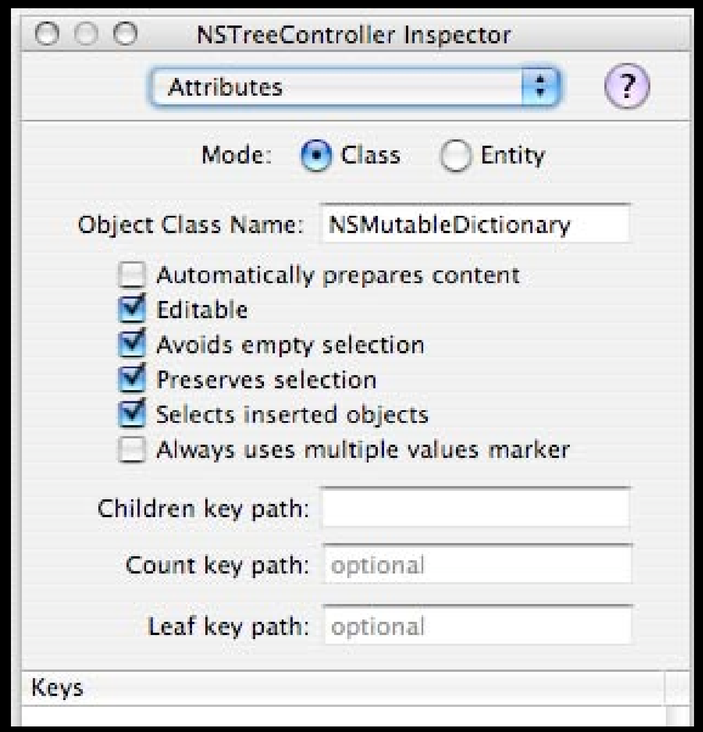
- Proxy objects for `selection` and `arrangedObjects`
- IB Actions are performed delayed

NSObjectController

- Superclass for NSArrayController, NSTreeController
- Single content object
- Useful in IB as a proxy for other objects
  - Master/Detail Interfaces (more later)

NSTreeController

- Content tree configured through key paths based on root objects
  - `arrangedObjects` proxy class private in Tiger
- Use with NSOutlineView/NSBrowser
  - NSTreeController/NSOutlineView like NSArrayController/NSTableView
  - `sortDescriptors` applied on each level



NSTreeController

Configuration

- Three Keypaths
  - `childrenKeyPath` (Required)
  - `isLeafKeyPath` (Optional)
  - `countKeyPath` (Optional)
- Giving the tree controller more information will provide better performance

NSTreeController

Children Key Path

- Tree Controller assumes all objects in your tree respond to the same children key path
- Required! Really!
- Use a category or extra accessor on model objects
  - If a class doesn't have children by design, use `isLeafKeyPath`

NSTreeController

Is Leaf Key Path

- Prevent NSTreeController from snooping for children
- Optional
- Affects how disclosure triangles appear in outline views
  - If (isLeaf) -> never shows disclosure triangle
  - If (!isLeaf) -> always shows disclosure triangle
  - If (`isLeafKeyPath` == nil)
    - If (number of children > 0) show disclosure triangle

NSTreeController

Count Key Path

- Lets you handle more of the tree contents yourself
  - Send KVO message when an objects child count changes
- Prevents add/insert/remove actions
- Affects how disclosure triangles appear in outline views
  - If (children > 0) show disclosure triangle

NSTreeController

IB Actions

- Beyond add/insert/remove
  - insertChild:(id)sender
  - addChild:(id)sender
- Special controller keys for enabling as well
  - canAddChild/canInsertChild

Demo

Ron Lue-Sang

http://developer.apple.com/wwdc2005/

Getting the Most Out of Cocoa Bindings

Andreas Wendker
Senior Manager
Xcode and Core Data Frameworks

http://developer.apple.com/wwdc2005/

Tips and Tricks for Advanced Bindings

- Master/detail interfaces
- Selection control bindings
- Target/argument bindings

Tips and Tricks for Advanced Bindings

- Master/detail interfaces
- Selection control bindings
- Target/argument bindings

Master/Detail Interfaces

Situation

- Two or more types of model objects
  - Related to each other
- User interface "partitioned", each part concentrating on one type of object
  - Objects shown in one part ("detail") based on selection in the other part ("master")

Master/Detail Interfaces

Examples

- Xcode's groups and files ➞ list view
- iTunes' library/playlists ➞ songs
- Mail's mailboxes ➞ messages
- CoreRecipes' groups ➞ recipes

Master/Detail Interfaces

Solution with Bindings

- Use two controllers (”master” and “detail”)
- Make content of detail controller dependent on selection in master controller
- Simply bind one of the `content` bindings of detail controller through the selection of master controller
  - Key path `@”selection.detailKey”`
- Adding/removing objects through detail controller will mutate relationship

Master/Detail Interfaces

Controller Content Bindings

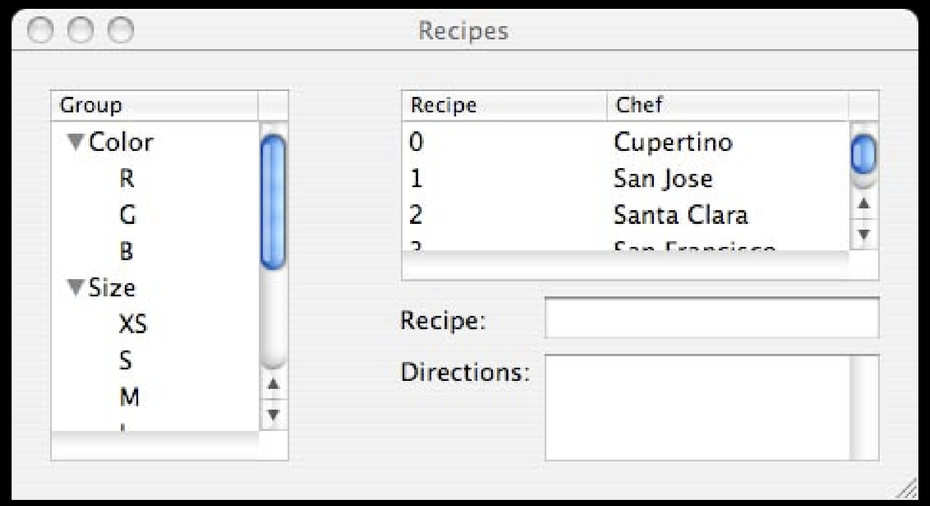
- Controllers offer several types of content bindings
- Use binding that matches your model object structure

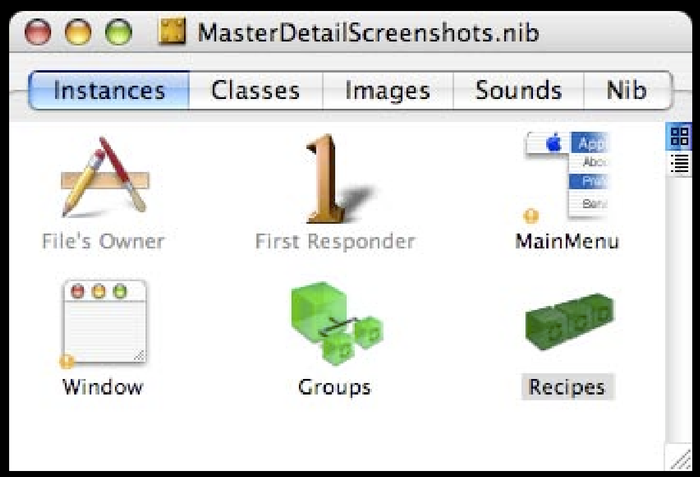
Binding	Expected Value	Use case
`contentArray`	`NSArray`	Ordered to-many relationship
`contentSet`	`NSSet`	Unordered to-many relationship
`contentObject`	`NSObject`	To-one relationship

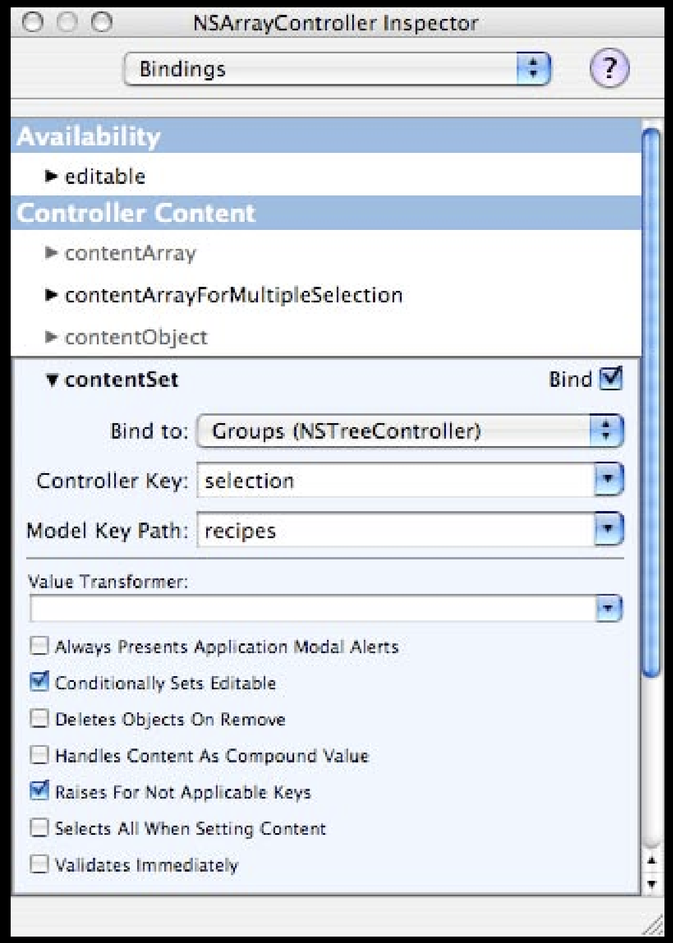
- Even unordered relationships need to be displayed ordered in user interface

Master/Detail Interfaces

Interface Builder Setup

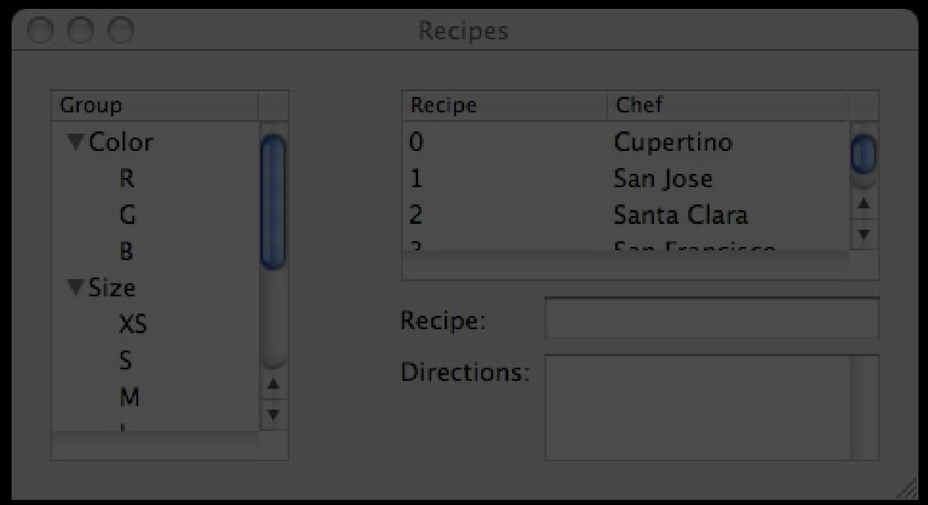


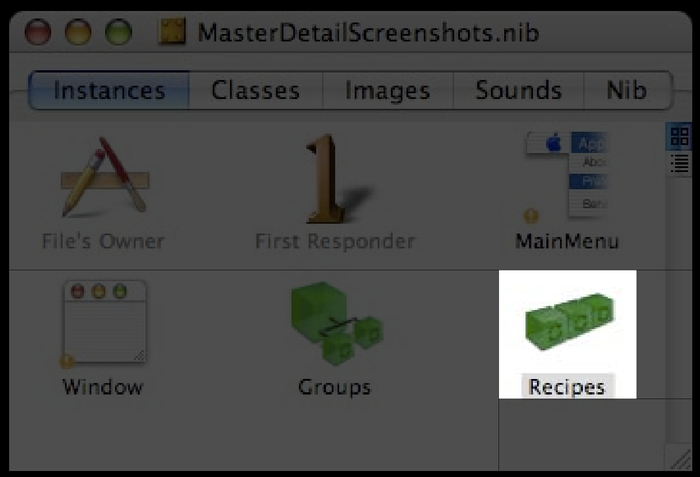


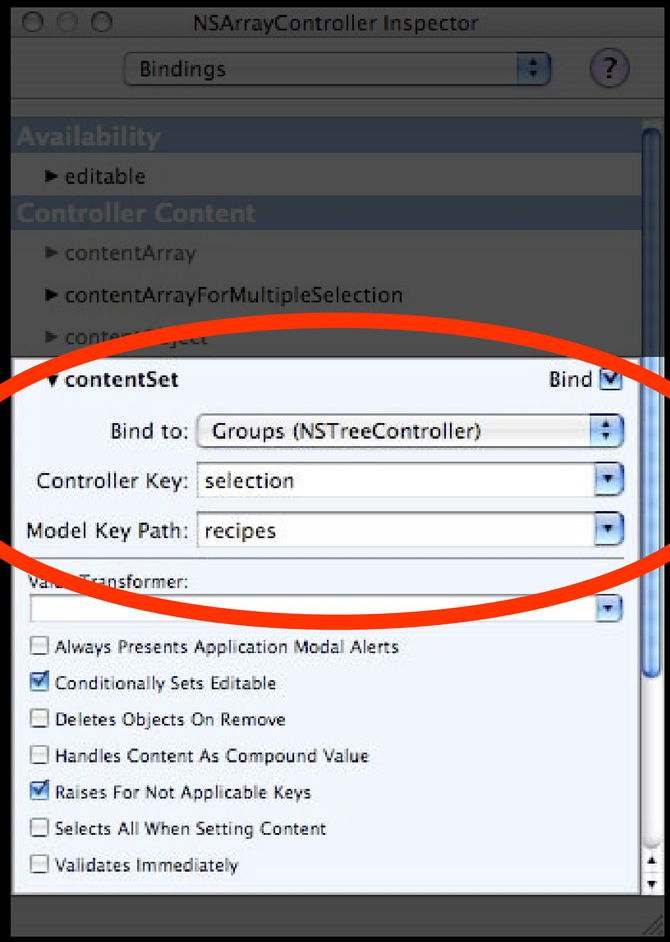


Master/Detail Interfaces

Interface Builder Setup

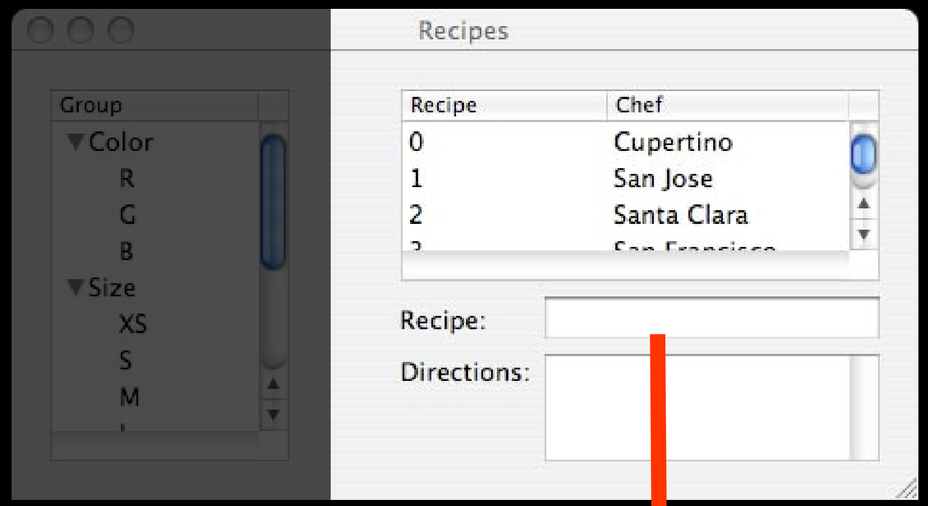




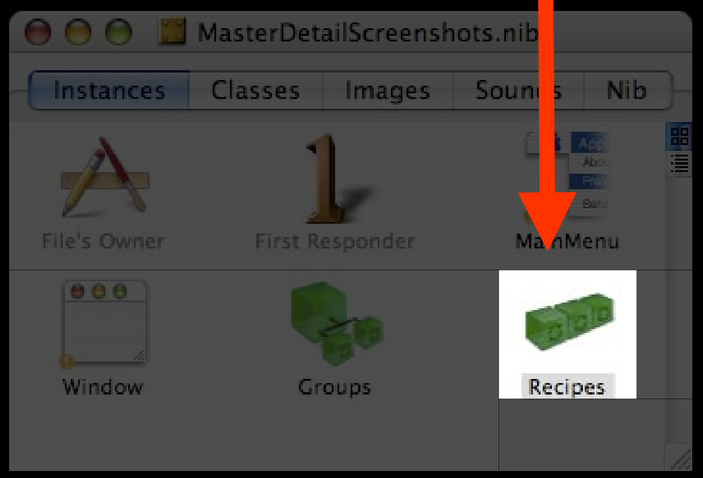


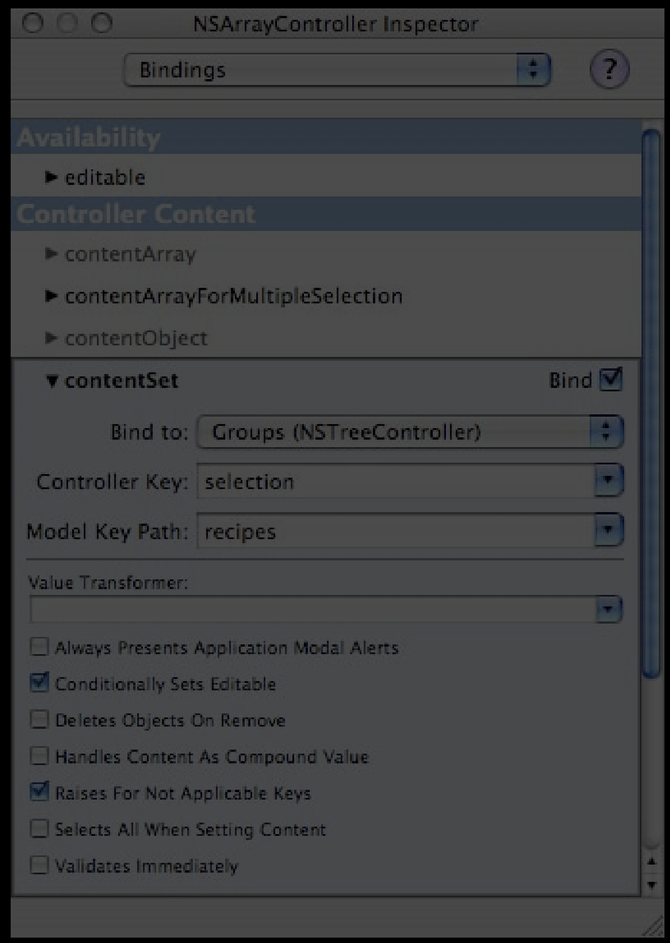
Master/Detail Interfaces

Interface Builder Setup



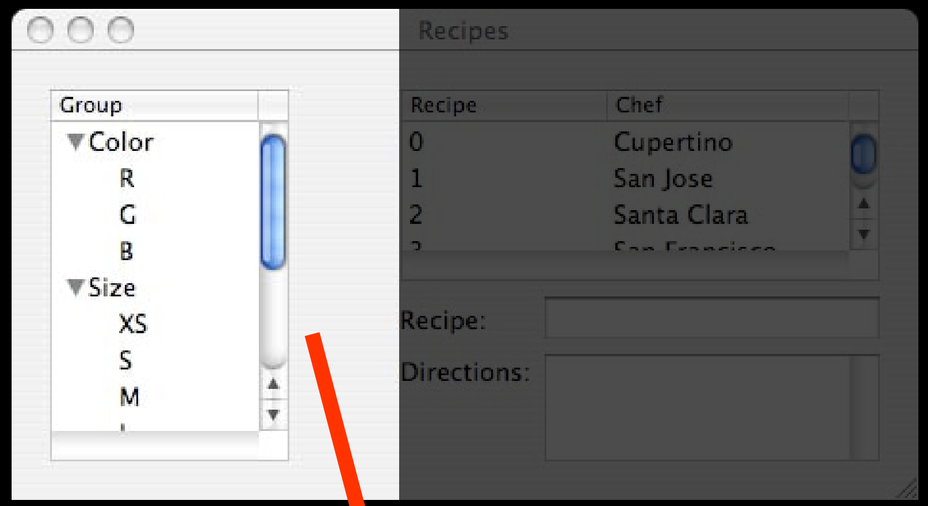
Additional Bindings



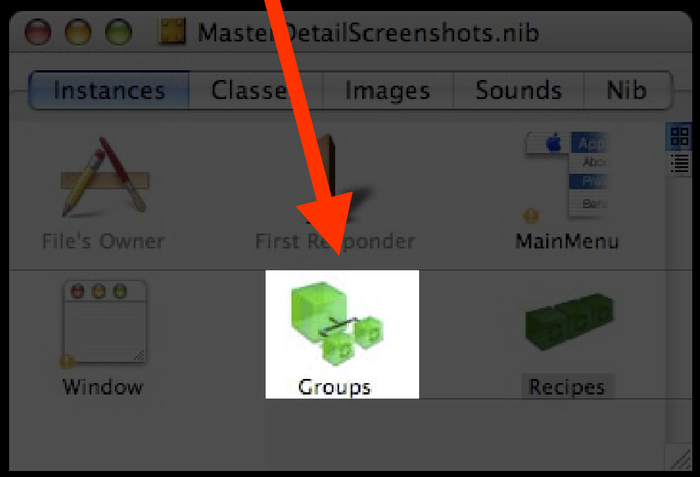


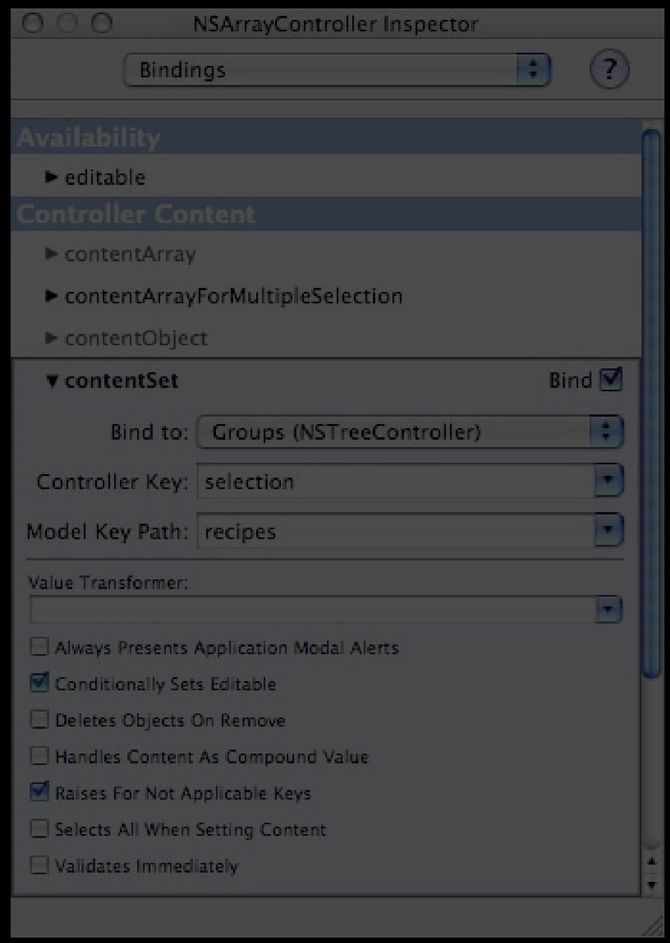
Master/Detail Interfaces

Interface Builder Setup



Additional Bindings





Master/Detail Interfaces

Multiple Selection in Master Controller

- Multiple selection in master controller: Value for content binding of detail controller is `NSMultipleSelectionMarker`
- By default, detail controller will be emptied
- Alternatively, detail controller will fall back to optional `contentArrayForMultipleSelection` binding
  - Allows to specify an alternative key path, value transformer, etc. for the multiple-selection case
  - Typically uses Key-Value Colding array operators

Master/Detail Interfaces

KVC Operators for Arrays and Sets

- NSArrays, controller array proxies offer variety of operators to combine objects from different types of relationships in arrays

Constant	Value
NSDistinctUnionOfArraysKeyValueOperator	@"distinctUnionOfArrays"
NSDistinctUnionOfObjectsKeyValueOperator	@"distinctUnionOfObjects"
NSDistinctUnionOfSetsKeyValueOperator	@"distinctUnionOfSets"
NSUnionOfArraysKeyValueOperator	@"unionOfArrays"
NSUnionOfObjectsKeyValueOperator	@"unionOfObjects"
NSUnionOfSetsKeyValueOperator	@"unionOfSets"

http://developer.apple.com/documentation/Cocoa/Conceptual/KeyValueCoding/Concepts/ArrayOperators.html

Master/Detail Interfaces

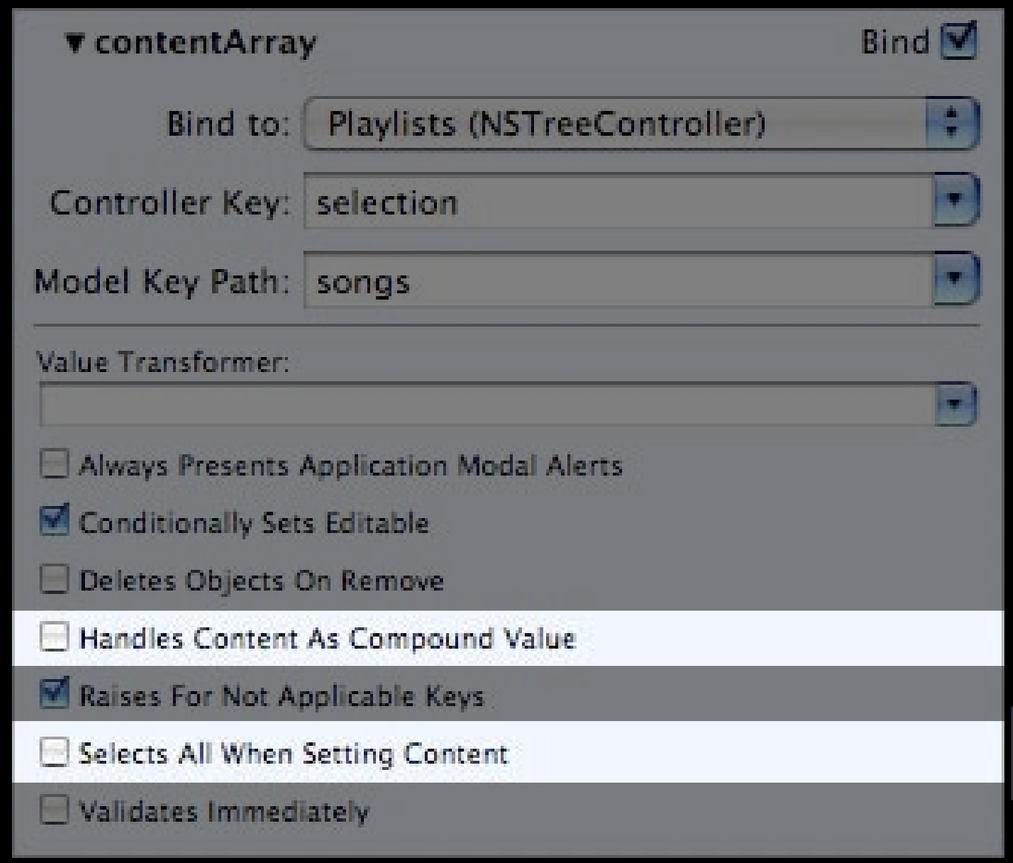
Multiple Selection in Master Controller

- Example: Recipes list, bound to Groups outline view (Recipes in unordered to-many relationships of Groups)

Detail Controller Binding	Key Path from Master Controller
contentSet	@”selection.recipes”
contentArrayForMultipleSelection	@”selection.@unionOfSets.recipes”

Master/Detail Interfaces

Additional Content Bindings Options



`NSHandlesContentAsCompoundValueBindingOption`

`NSSelectsAllWhenSettingContentBindingOption`

Master/Detail Interfaces

Handles Content as Compound Value

- Typically used with a value transformer on `content` binding
- `NSHandlesContentAsCompoundValueBindingOption` prevents controller from using standard KVC array manipulation method to mutate content relationship
  - Bypass `-mutableArrayValueForKey:` mechanism
  - Single KVC get/set operations, value transformer applied on the way
- Examples
  - Unarchive entire array of objects from single NSData object
  - Create array of individual “line” objects from dictionary with key/value pairs

Master/Detail Interfaces

Selects All when Setting Content

- Typically used with controllers for inspector panels
  - Selection is pre-determined by the master controller
  - Always edit all objects received from master controller
- `NSSelectsAllWhenSettingContentBindingOption` makes sure that all content objects are selected when controller is populated
- Example key path in inspector of a document based application:
  `mainWindow.delegate.controller.selectedObjects`
  (bound to NSApplication instance)

Tips and Tricks for Advanced Bindings

- Master/detail interfaces
- Selection control bindings
- Target/argument bindings

Selection Control Bindings

Situation

- Manage selected item in NSPopUpButton
  - Used for relationship manipulation (esp. Core Data)
  - Typically based on represented object, tag, index, or display value
- Populate control with available choices
  - Manually with a fixed list in Interface Builder
  - Dynamically, determined at runtime

Selection Control Bindings

Solution with Bindings

- Use `selected` bindings to manage selected item
- Use `content` bindings to populate controls
- Bindings work with standalone controls and cells in table views
  - Can populate pop-ups in each row of table view with different content

Selection Control Bindings

Selected Bindings

- `selected` bindings work both with manually defined content (entered in Interface Builder) or with dynamic content

Binding	Basis
`selectedObject`	Represented object of selected item
`selectedTag`	Tag of selected item
`selectedIndex`	Index of selected item
`selectedValue`	Display value of selected item (title)

Selection Control Bindings

Content Bindings

- Several `content` bindings available
  - `content`
  - `contentValues`
  - `contentObjects`
- Choose depending on how model objects are organized

Selection Control Bindings

Content Bindings

- Case one—simple array of NSStrings
  - Bind `contentValues` binding to string array
  - Just display values, no represented objects (all nil)
  - Note that KVC operations on `NSArrays` returns `NSArrays`
    - Value for key path `@"someObjects.name"` is an array of strings
  - Can bind to array controller with `@"arrangedObjects.name"`

Selection Control Bindings

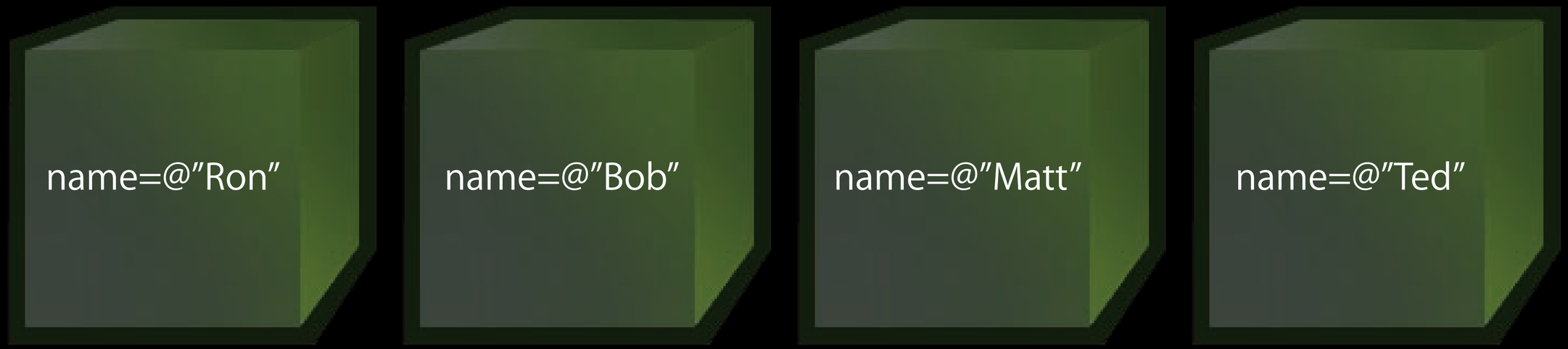
Content Bindings

- Case two—array of represented objects
  - Bind `content` binding to represented objects array
    - Display value will be `-description` of objects
  - Can use `selectedObject` binding to manage selection based on choices available in `content` array

Selection Control Bindings

Content Bindings

- Case three—array of represented objects with dedicated display value

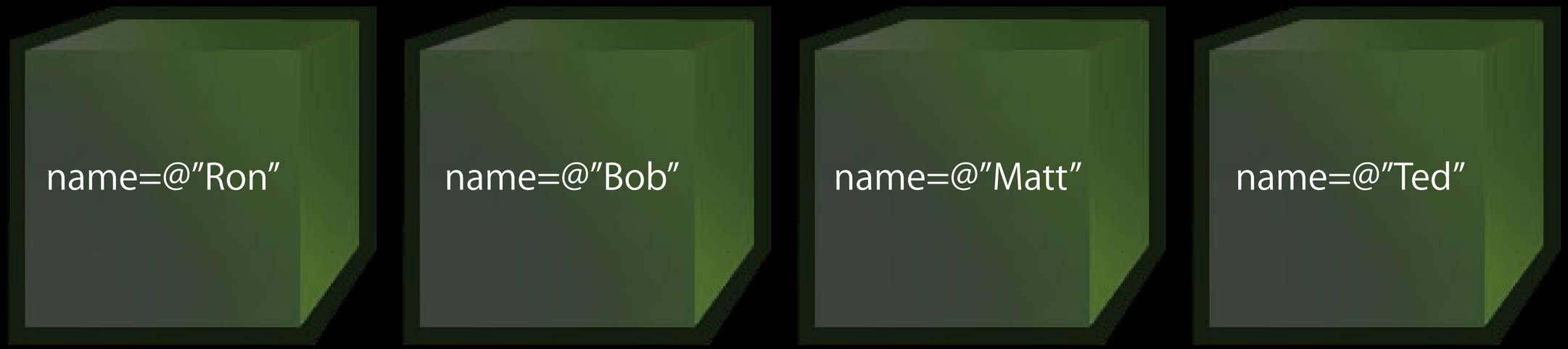


- Bind `content` binding to array of represented objects
- Bind `contentValues` binding to display value through represented objects
- Can use `selectedObject` binding to manage selection based on choices available in `content` array, still get a nice display value

Selection Control Bindings

Content Bindings

- Case three—array of represented objects with dedicated display value



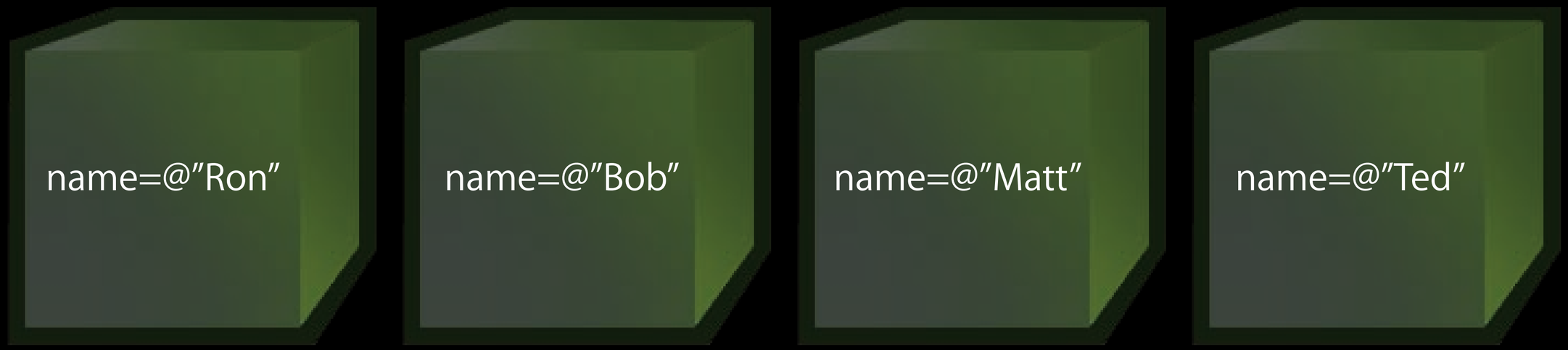
- Example: populate control with all objects from a Person array controller, use @"name" as display value

Binding	Key Path
`content`	`@"arrangedObjects"`
`contentValues`	`@"arrangedObjects.name"`

Selection Control Bindings

Content Bindings

- Case three—array of represented objects with dedicated display value

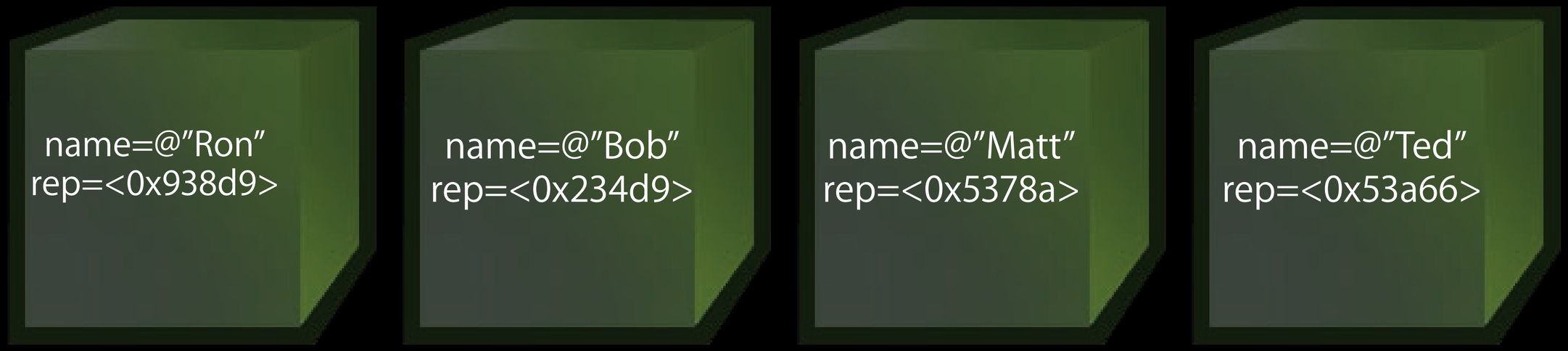


  - Note: display values need to be accessible through represented objects
    - Bind both `content` and `contentValues` to same object
    - Key path for `content` needs to be root of key path for `contentValues`

Selection Control Bindings

Content Bindings

- Case four—array of base objects with dedicated properties for represented objects and display values

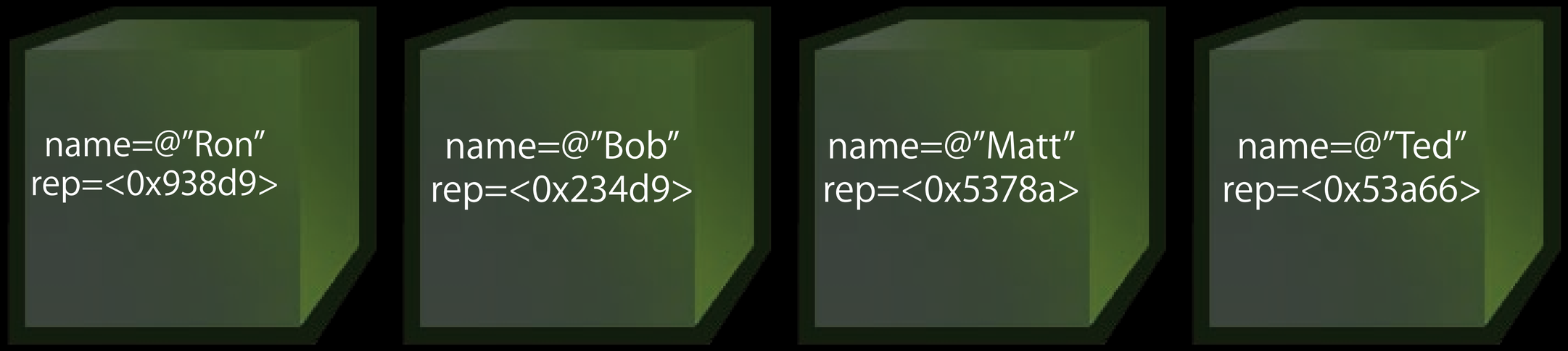


  - Typically NSDictionary with two key/value pairs
  - Bind `content` binding to array of base objects
  - Bind `contentValues` binding to display value through base objects
  - Bind `contentObjects` binding to represented objects through base objects

Selection Control Bindings

Content Bindings

- Case four—array of base objects with dedicated properties for represented objects and display values



  - Can use `selectedObject` binding to manage selection based on choices available in `contentObjects`, still get a nice display value through `contentValues`
  - Note: display values and represented objects need to be accessible through single array of base objects

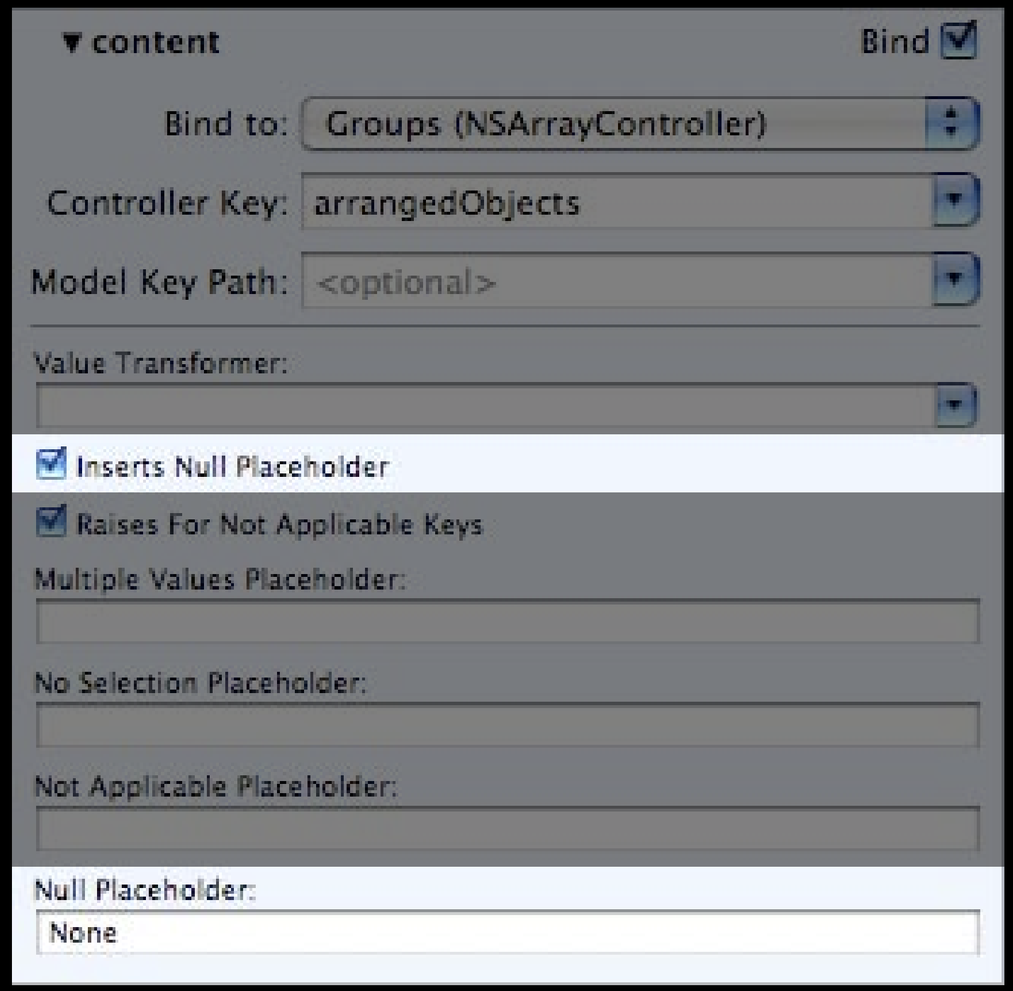
Selection Control Bindings

Content Bindings Combinations

Bindings	Represented Objects, `selectedObject` binding	Display Values
`contentValues`	N/A	Values in `contentValues` array
`content`	Objects in `content` array	Simply `-description` of objects in `content` array
`content` & `contenValues`	Objects in `content` array	Values in `contentValues` array
`content` & `contentValues` & `contentObjects`	Objects in `contentObjects` array	Values in `contentValues` array

Selection Control Bindings

Additional Content Bindings Options



`NSInsertsNullPlaceholderBindingOption`

`NSNullPlaceholderBindingOption`

Selection Control Bindings

NSMatrix Bindings

- `NSMatrix` has same `content` bindings as `NSPopUpButton`
- Depending on mode, different `selected` bindings
  - Radio mode (single selection)
    - `selectedObject`
    - `selectedTag`
    - `selectedIndex`
    - `selectedValue`
  - List/Highlight mode (multiple selection)
    - `selectedObjects`
    - `selectedValues`

Tips and Tricks for Advanced Bindings

- Master/detail interfaces
- Selection control bindings
- Target/argument bindings

Target/Argument Bindings

Situation

- Invoke arbitrary methods, potentially with multiple arguments, on arbitrary objects
- Without bindings, Interface Builder offers no convenient way
  - Target/action methods only, limited to sender argument
  - Need outlets for all objects to read arguments from

Target/Argument Bindings

Solution with Bindings

- Use `target`/`argument` binding combinations on `NSButton`
  - `Target` binding specifies receiver of method
    - Can make use of `@"self"` as key path
  - Selector name entered in options section of bindings inspector
    - One `argument` binding for each argument of method invocation
      - Additional enumerated `argument2`, `argument3`, etc. bindings become available while binding arguments
      - As with all bindings, additional flexibility through value transformers
- Use `doubleClickTarget`/`doubleClickArgument` bindings on `NSTableView`
  - Work just the same way

Target/Argument Bindings

Examples

- Invoke method

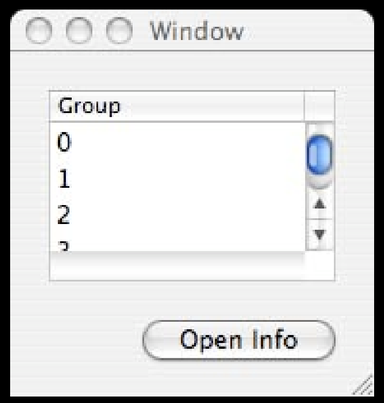
```
-(void)openInfoPanelsForGroups:(NSArray *)groups
```

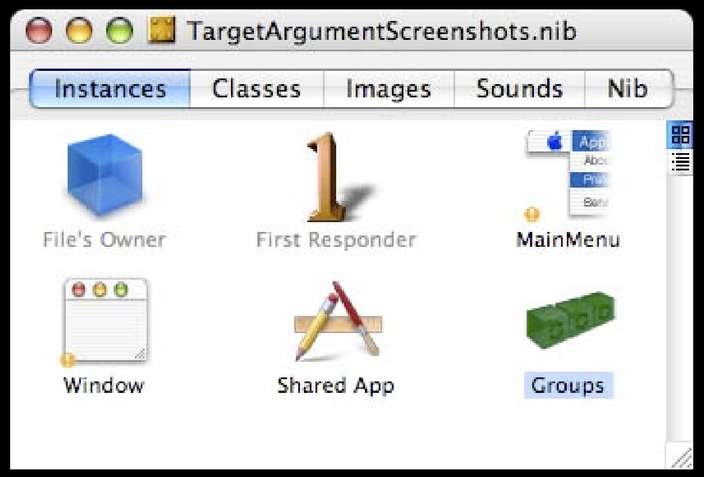
on NSApplication delegate,
with objects selected in a Groups table view

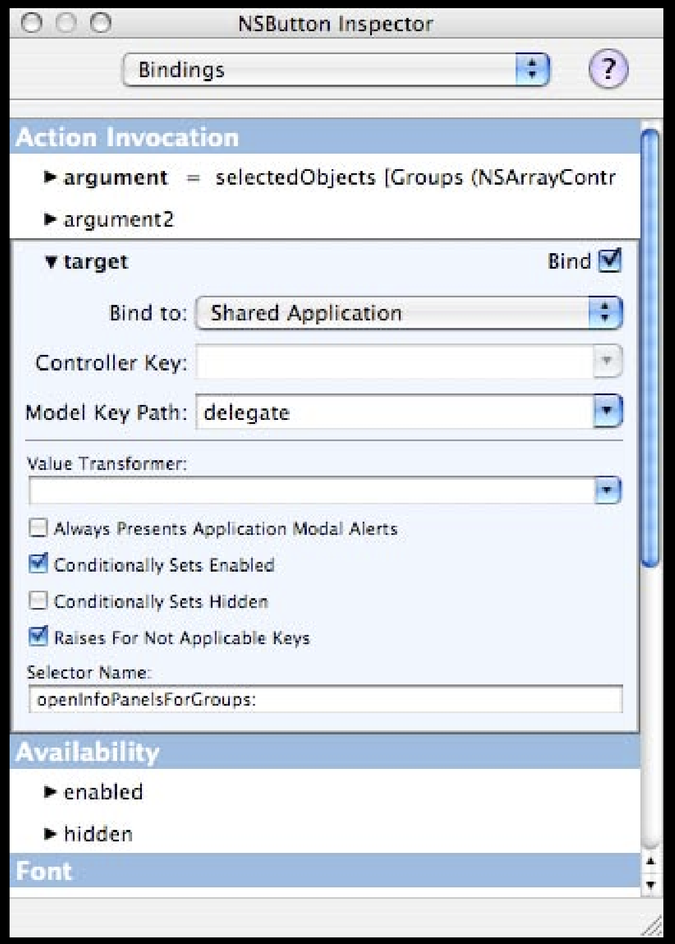
Binding	Object Bound To	Key Path
`target`	Shared application instance	`delegate`
`argument`	Array controller for Groups objects	`selectedObjects`

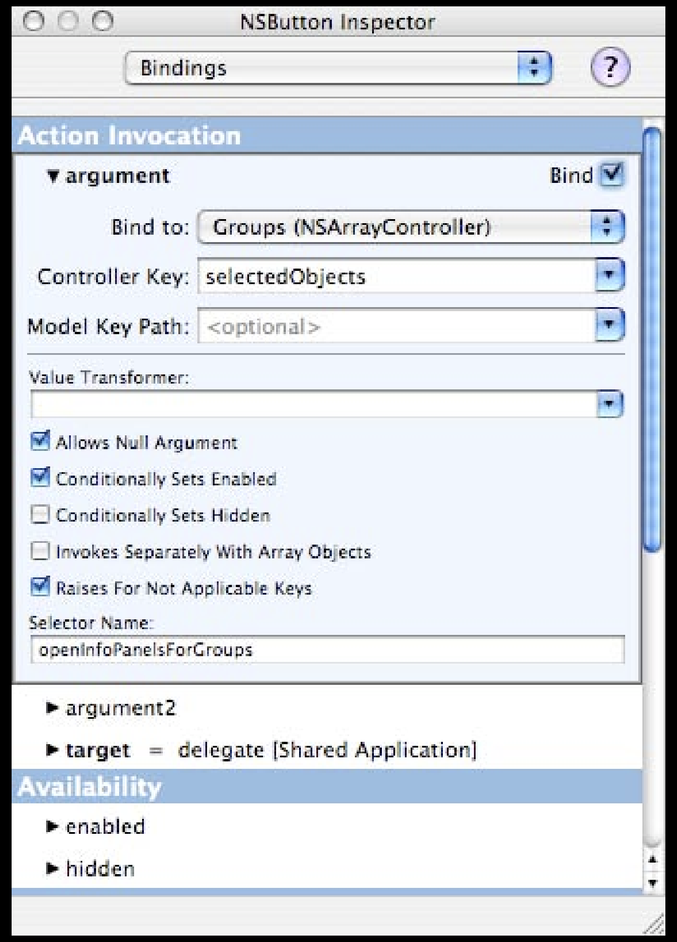
Target/Argument Bindings

Interface Builder Setup



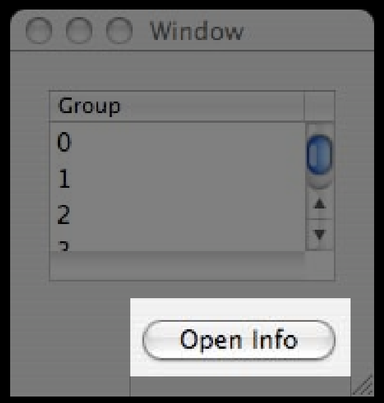


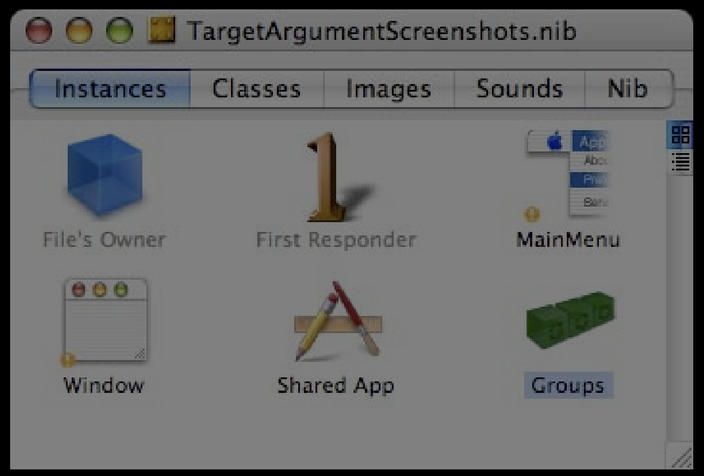


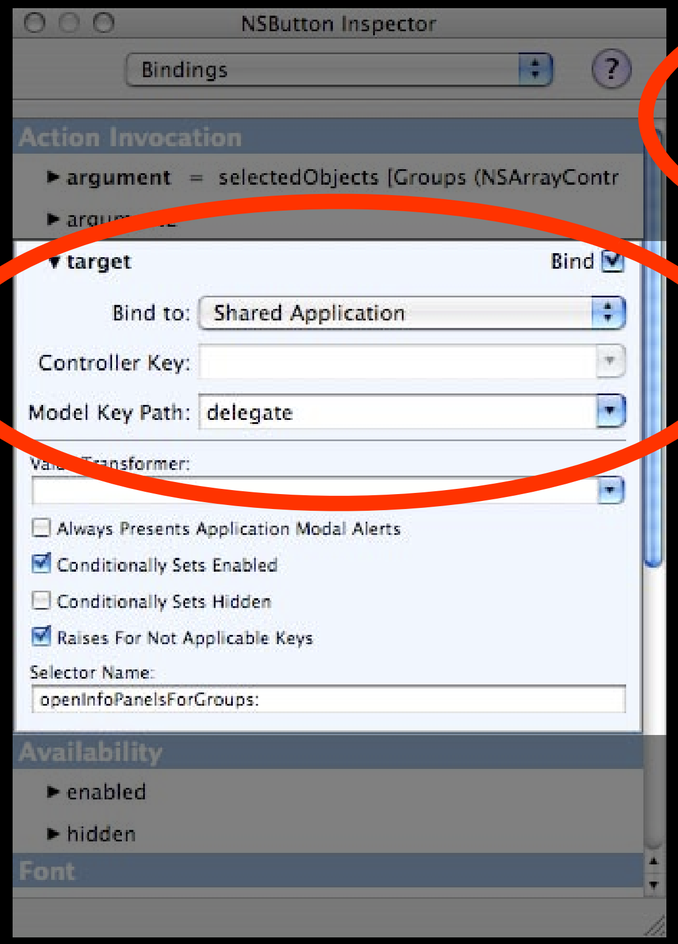


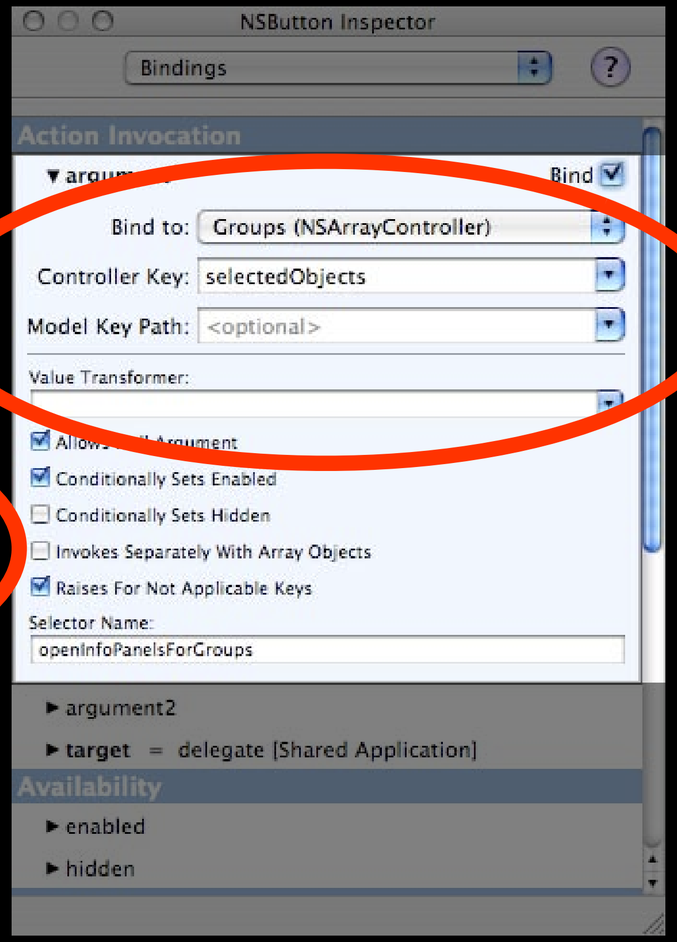
Target/Argument Bindings

Interface Builder Setup









Target/Argument Bindings

Examples

- Invoke method

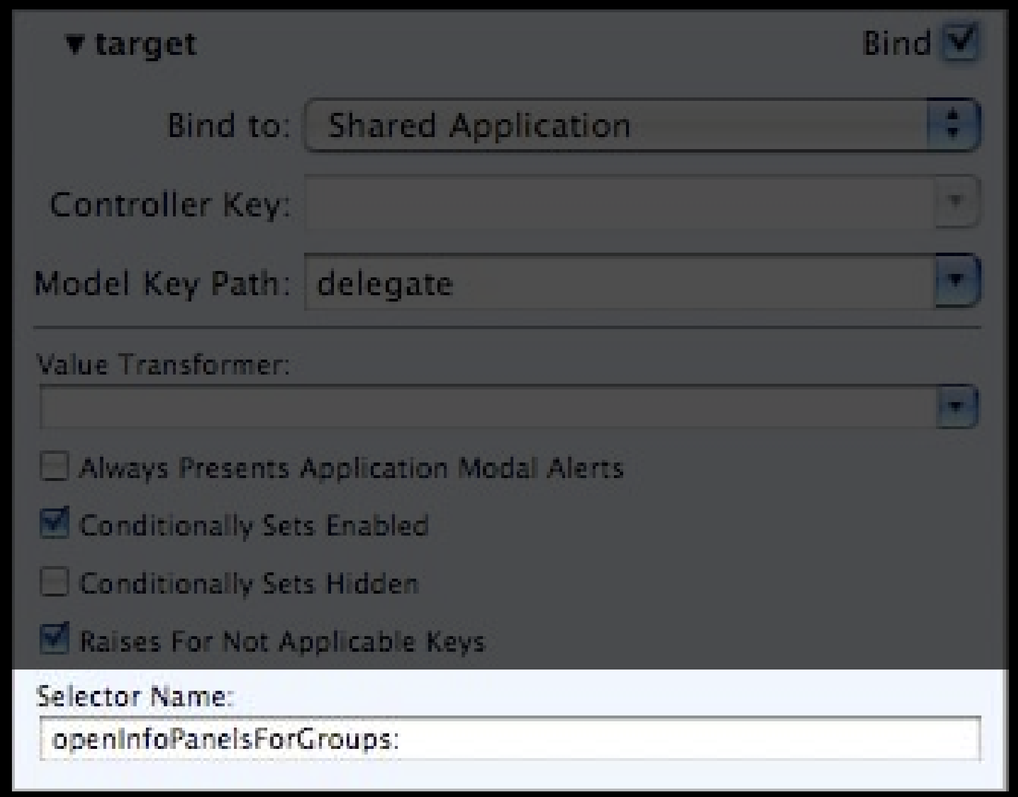
```
-(void)addKeywords:(NSArray *)strings
toPictures:(NSArray *)objects showProgressIndicator:(BOOL)flag
```

on File's Owner,
with keywords (NSStrings) selected in one table view,
with Picture objects selected in another table,
and with flag read from user defaults

Binding	Object Bound To	Key Path
`target`	File's Owner	`self`
`argument`	Array controller for Keyword objects	`selectedObjects.name`
`argument2`	Array controller for Picture objects	`selectedObjects`
`argument3`	Shared user defaults controller	`values.YourUserDefault`

Target/Argument Bindings

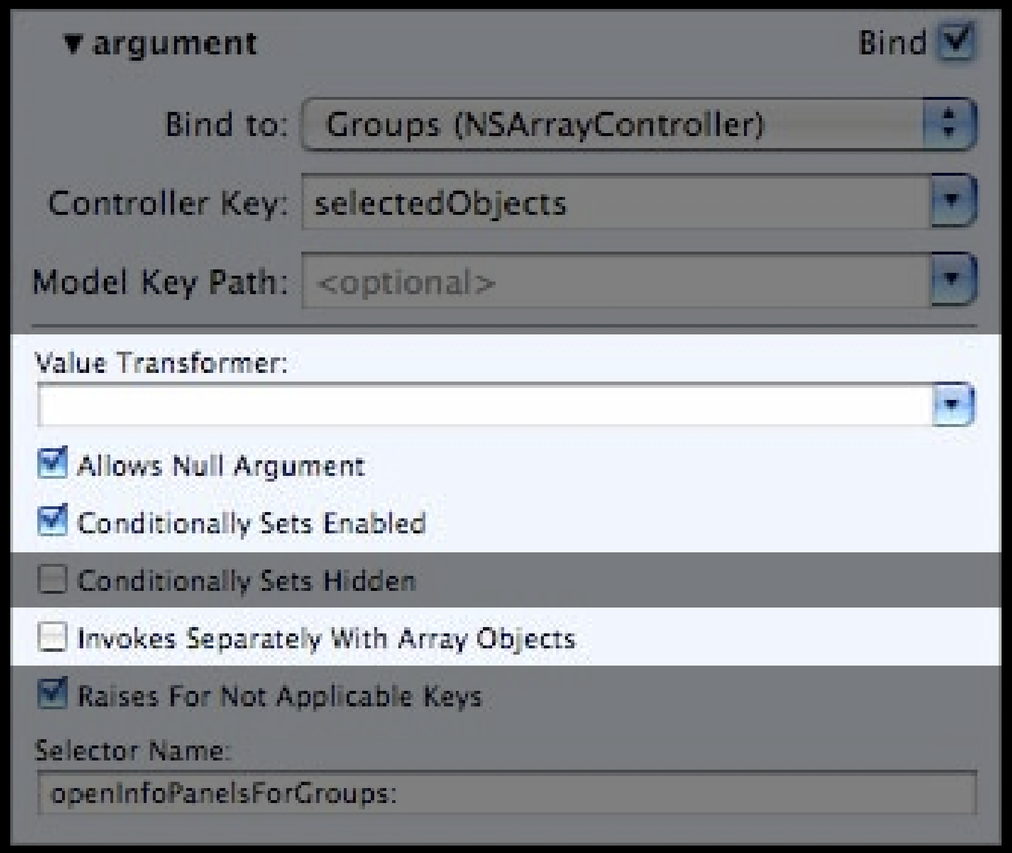
Selector Name Required




```
NSSelectorNameBindingOption
```

Target/Argument Bindings

Additional Argument Bindings Options



- `NSValueTransformerNameBindingOption`
- `NSAllowsNullArgumentBindingOption`
- `NSConditionallySetsEnabledBindingOption`
- `NSInvokesSeparatelyWithArrayObjectsBindingOption`

Demo

Ron

http://developer.apple.com/wwdc2005/

Getting The Most Out of Cocoa Bindings

Ron Lue-Sang
Engineer
Cocoa Data/Core Bindings

http://developer.apple.com/wwdc2005/

Controllers and Core Data

NSControllers and Core Data

NSObject/Array/TreeController Features

- New “Entity” mode (instead of “Class” mode)
- contentSet binding mentioned earlier
- Controllers fetch content from a NSManagedObjectContext

NSControllers and Core Data

ContentSet binding

- Available for NSArrayController and NSTreeController
- Purpose
  - Core Data relationships are unordered (NSSets)
  - Content sets are converted into content arrays/trees
- Controller will tell you if you bound the wrong thing

```
2005-05-27 14:13:39.077 spiff[7025] *** NSRunLoop ignoring
exception 'Cannot create NSArray from object
<_NSFaultingMutableSet: 0x3818f0> () of class
_NSFaultingMutableSet - consider using contentSet binding
instead of contentArray binding' that raised during posting
of delayed perform with target 385350 and selector
'invokeWithTarget:'
```

NSControllers and Core Data

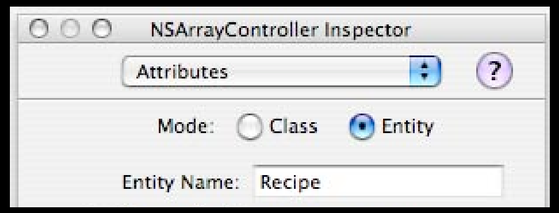
Configuring Controllers to Fetch Content

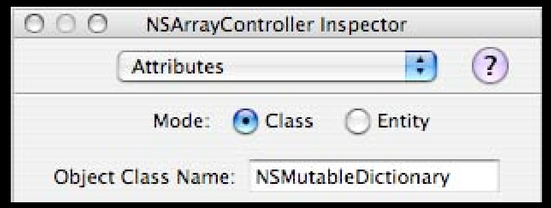
- Set an entity name in the controller
- Provide the controller with a managed object context
- Optional
  - Set the fetch predicate
  - Enable live updates

NSControllers and Core Data

1) Provide an entity name (Required)

- In IB, switch to Entity mode
- Set the name of the entity you want to fetch
- Instances of <entity> will be created during add/insert actions

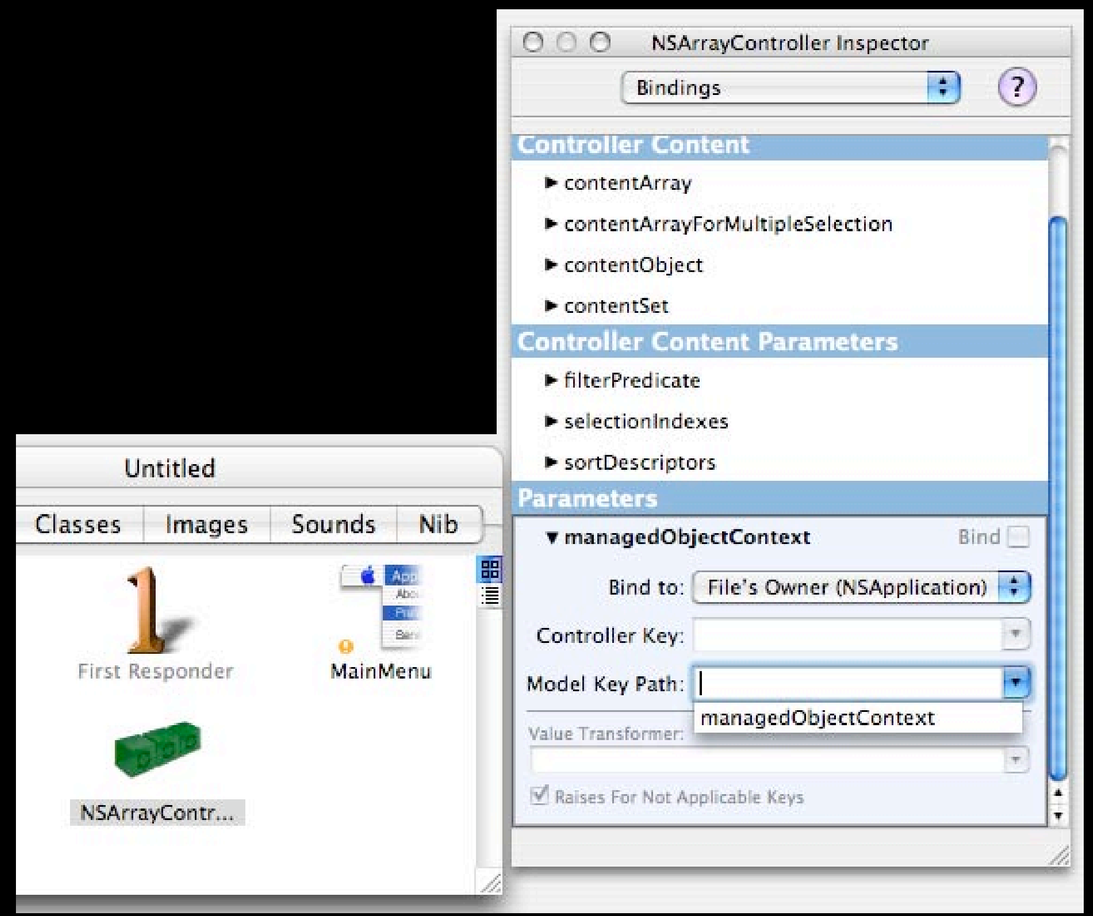




NSControllers and Core Data

2) Provide controller with a NSManagedObjectContext (Required)

- Bind controller's `managedObjectContext`



NSControllers and Core Data

When controllers create new managed objects…

- New objects inserted into the controller's NSManagedObjectContext
- By default, new objects are assigned to the first writable store
- By default, all fetches performed over all stores of the NSManagedObjectContext's coordinator
- Consider subclassing to override…

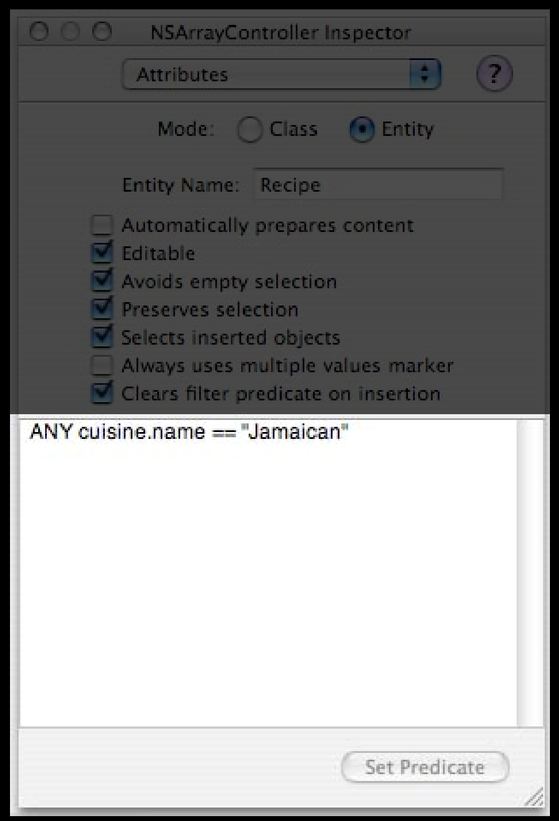
```
- (id)newObject
```

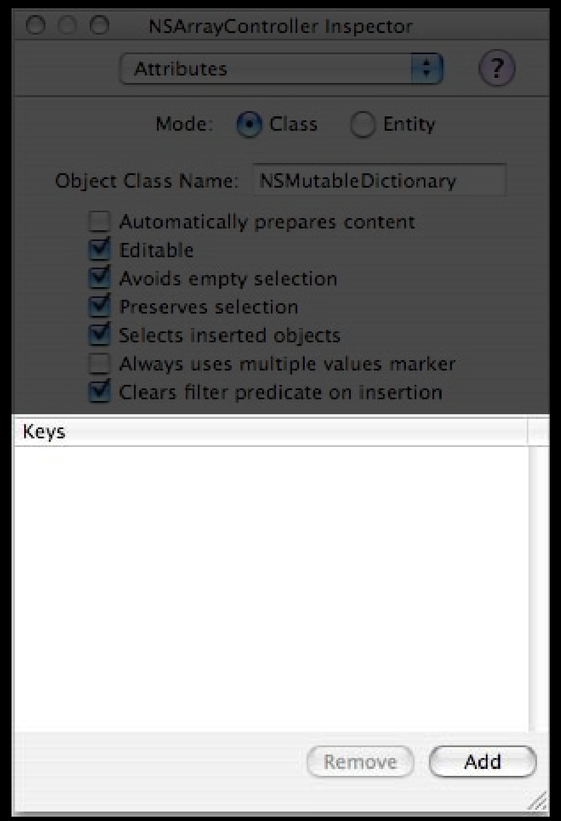
```
- (BOOL)fetchWithRequest:(NSFetchRequest *)fetchRequest
                   merge:(BOOL)merge
                   error:(NSError **)error
```

NSControllers and Core Data

3) Set Fetch Predicate (optional)

- Only available in Entity Mode

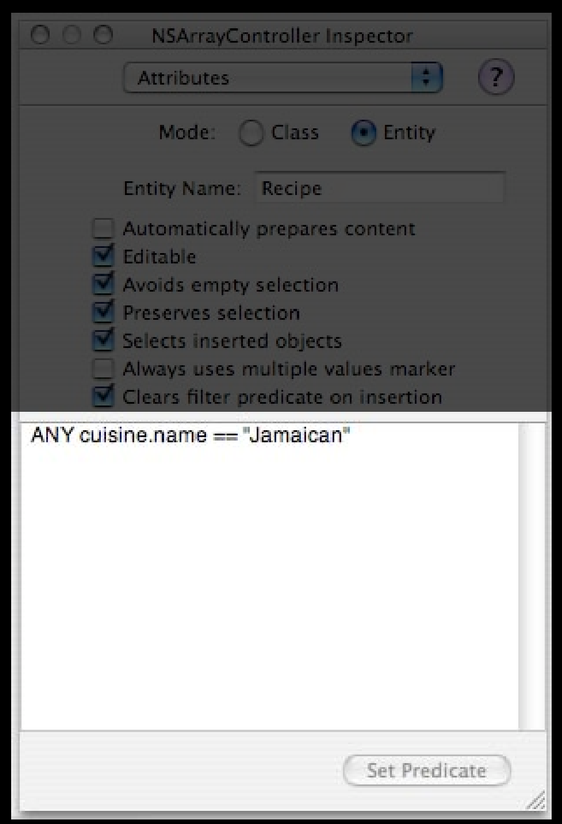


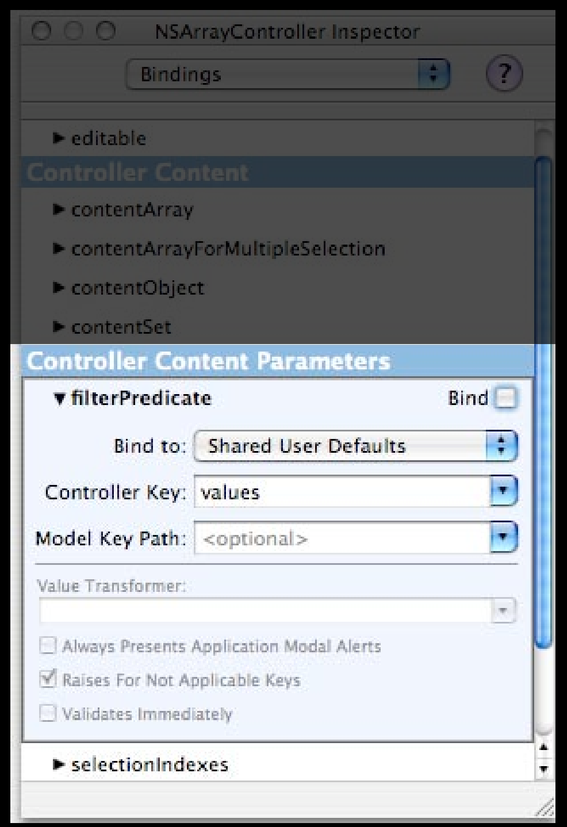


NSControllers and Core Data

3) Set Fetch Predicate (optional)

- Describes which objects of the defined entity should become part of the controller content
- fetchPredicate != filterPredicate

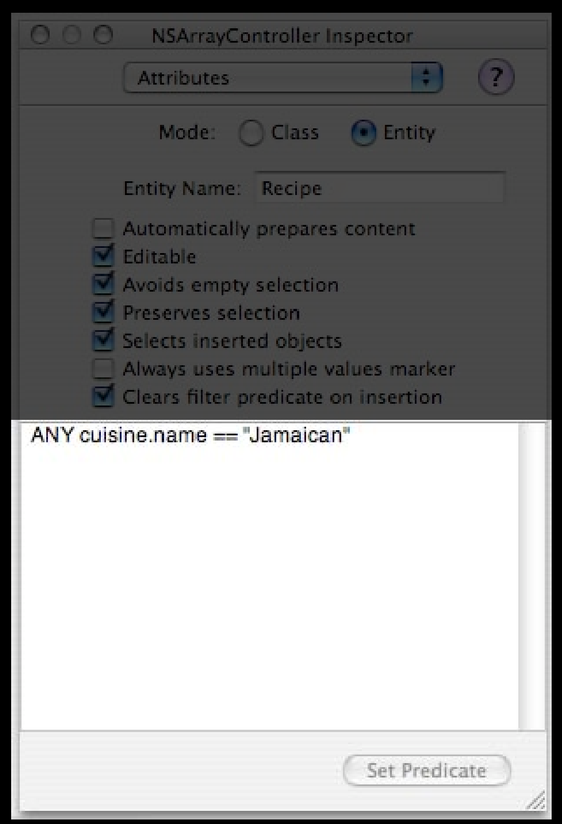


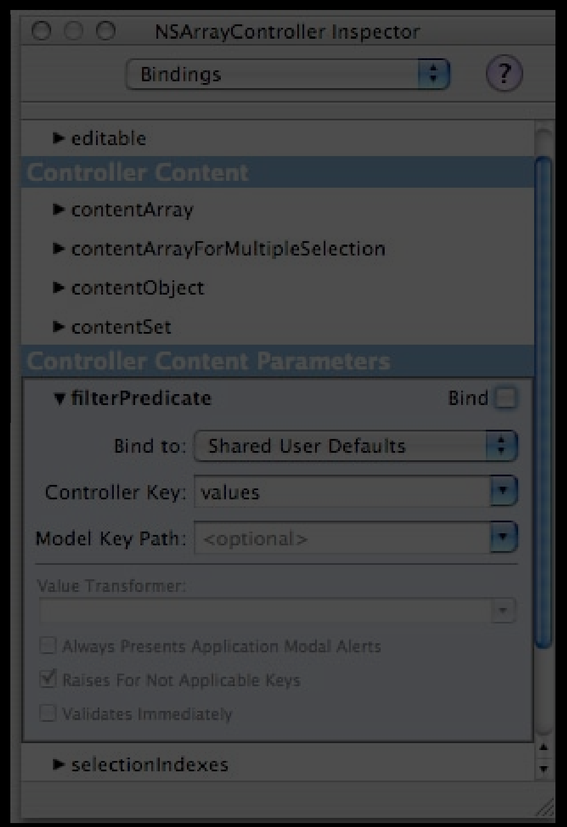


NSControllers and Core Data

fetchPredicate != filterPredicate

- `fetchPredicate` affects which objects become part of the controller content

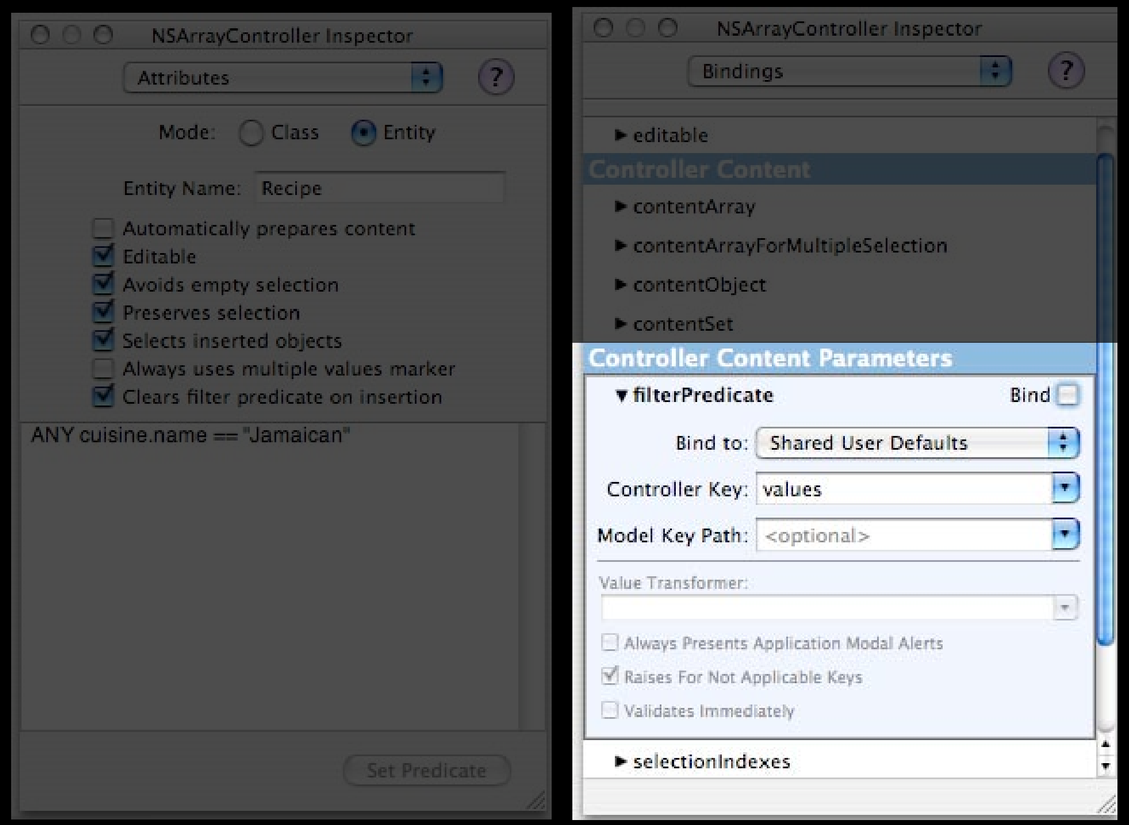




NSControllers and Core Data

fetchPredicate != filterPredicate

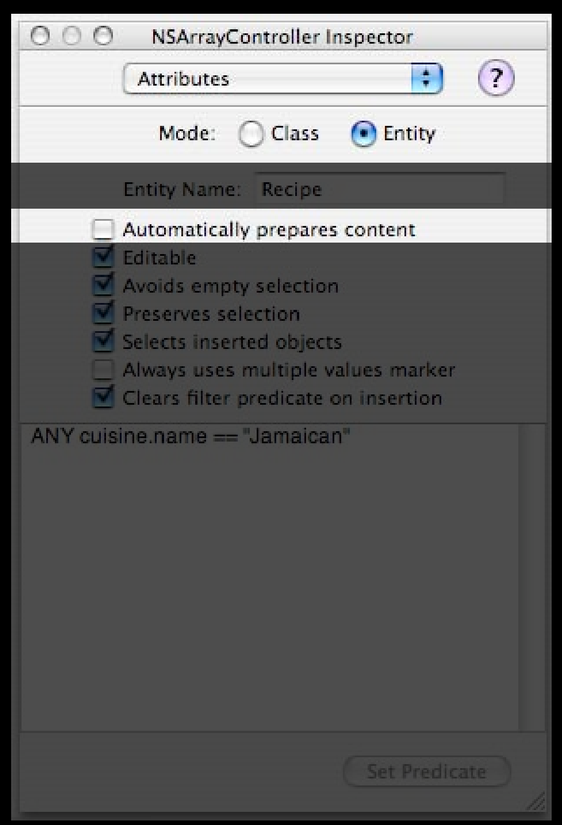
- `filterPredicate` affects which objects become part of the arrangedObjects
- `arrangedObjects` is always a subset of objects in content

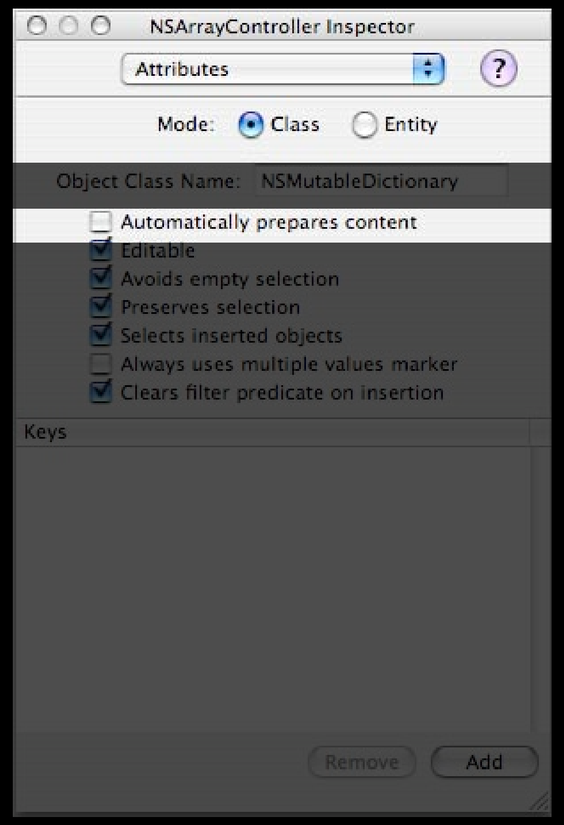


NSControllers and Core Data

4) Enable live updates (optional)

- Inserting/Deleting/Updating objects in the NSManagedObjectContext results in the controller updating its content
- To enable, check “Automatically prepares content”

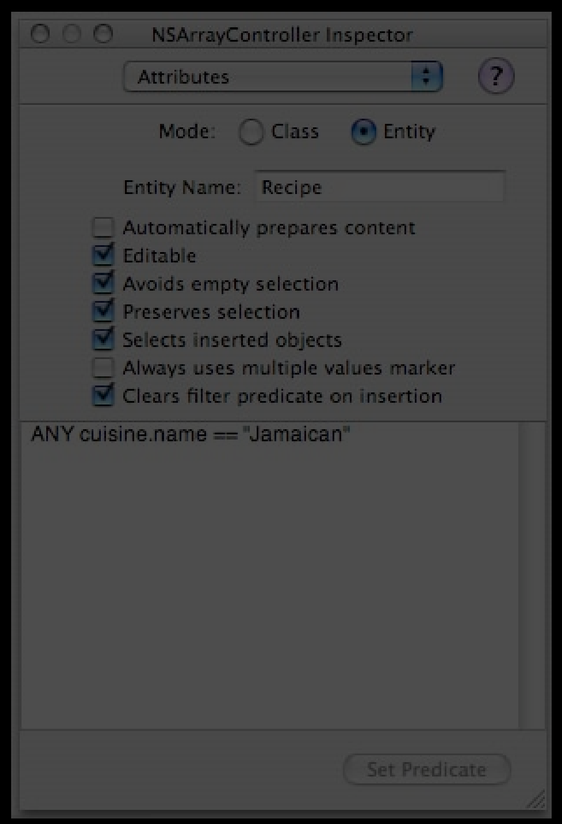


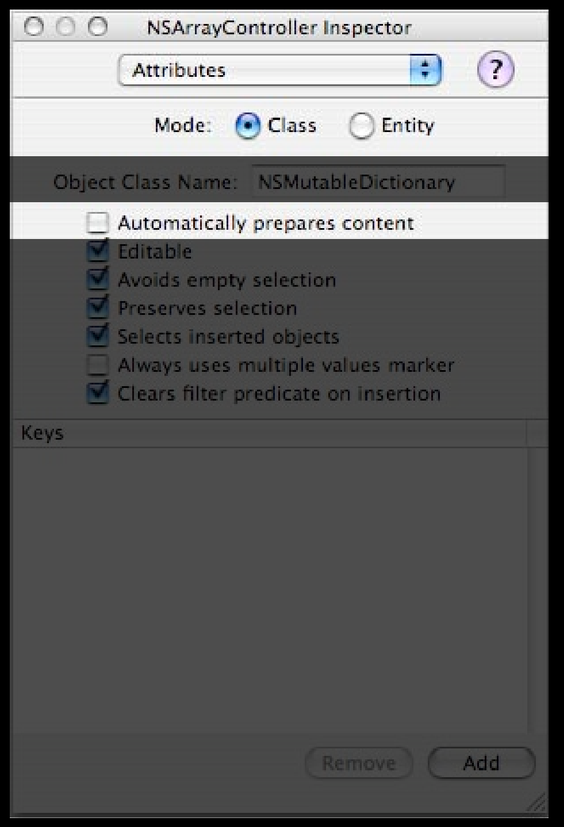


NSControllers and Core Data

Automatically prepares content

- In class mode, in awakeFromNib controller will create one content object if its content is nil

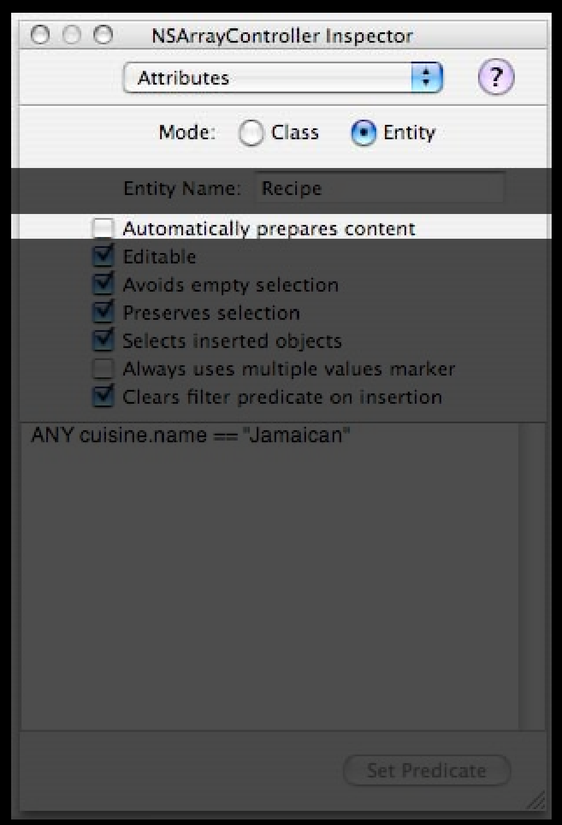


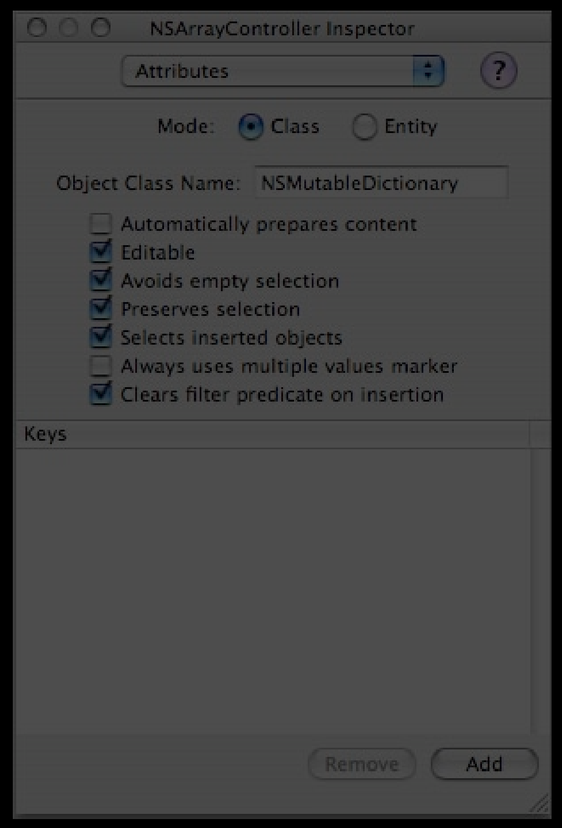


NSControllers and Core Data

Automatically prepares content

- In entity mode, in awakeFromNib controller will fetch with its NSManagedObjectContext
- Any context notifications result in the controller updating its content—less work for you!





NSControllers and Core Data

NSObject/Array/TreeController Features

- New “Entity” mode (instead of “Class” mode)
- `contentSet` binding mentioned earlier
- Controllers fetch content from a NSManagedObjectContext

Bindings: More Stuff

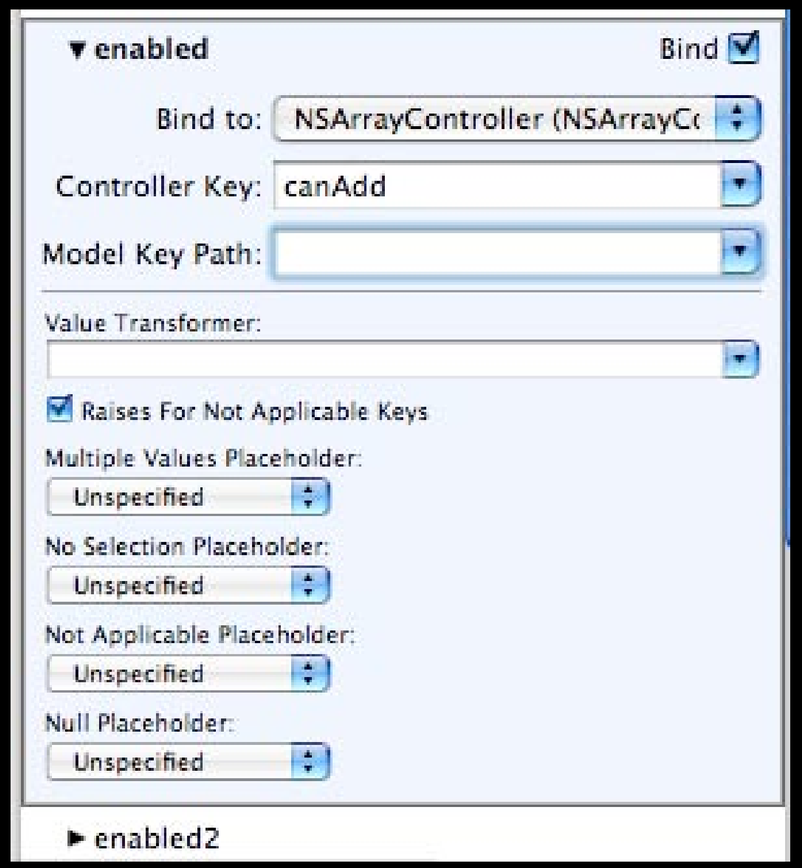
Bindings: More Stuff

- Enabled/Hidden/Editable bindings
- Error presentation details
- Debugging Tips

Bindings: Availability Bindings

Enabled/Hidden/Editable bindings

- "Enumerated" bindings
  - Bind enabled, enabled2 comes up



Bindings: Availability Bindings

Enabled/Hidden/Editable bindings

- Enabled/Editable logic
  - Values for all enabled/editable bindings (enabled, enabled2, enabled3, etc.) ANDed together
- Hidden logic
  - Values for all hidden bindings (hidden, hidden2, hidden3, etc.) ORed together

Bindings: Error Presentation

When committing edits fails

- Bound views can present errors as sheets
- NSEditor category on NSObject
- Callback based commitEditing

```
- (void)commitEditingWithDelegate:(id)delegate

  didCommitSelector:(SEL)didCommitSelector

  contextInfo:(void *)contextInfo
```

- Bindings use the new presentError API

Bindings: Debugging

- Most common error is incorrect keypath

```
2005-05-26 17:17:07.073 spiff[6644] *** NSRunLoop ignoring exception
'[<NSManagedObject 0x3bc4b0> valueForUndefinedKey:]: this class is not
key value coding-compliant for the key mame.' that raised during posting
of delayed perform with target 3bac10 and selector 'invokeWithTarget:'
```

Bindings: Debugging

- Another common error message: forgetting to set/bind a controller's managedObjectContext

```
2005-05-26 17:16:08.600 spiff[6638] *** NSRunLoop ignoring exception
'Cannot perform operation without a managed object context' that raised
during posting of delayed perform with target 3671d0 and selector
'invokeWithTarget:'
```

Bindings: Debugging

- NSBindingDebugLogLevel = 1
  - Watch the run/debug log
  - You'll only see a log when `NSUndefinedKeyException` is raised

```
2005-05-27 14:23:54.592 spiff[7041] Error setting value
for key path selection.name of object <NSArrayController:
0x357470>[entity: Recipe, number of selected objects: 1]
(from bound object <NSTextField: 0x3357e0>): I just felt
like doing something bad
```

Bindings: Debugging

- NSBindingDebugLogLevel = 1
  - Watch the run/debug log
  - You'll only see a log when `NSUndefinedKeyException` is raised

```
2005-05-27 14:23:54.592 spiff[7041] Error setting value for key
path selection.name of object <NSArrayController: 0x357470>
[entity: Recipe, number of selected objects: 1] (from bound
object <NSTextField: 0x3357e0>): I just felt like doing
something bad
```

Bindings: Debugging

- NSBindingDebugLogLevel = 1
  - Watch the run/debug log
  - You'll only see a log when `NSUndefinedKeyException` is raised

```
2005-05-27 14:23:54.592 spiff[7041] Error setting value for key
path selection.name of object <NSArrayController: 0x357470>
[entity: Recipe, number of selected objects: 1] (from bound
object <NSTextField: 0x3357e0>): I just felt like doing
something bad
```

Bindings: Debugging

- NSBindingDebugLogLevel = 1
  - Watch the run/debug log
  - You'll only see a log when `NSUndefinedKeyException` is raised

```
2005-05-27 14:23:54.592 spiff[7041] Error setting value for key path
selection.name of object <NSArrayController: 0x357470>[entity: Recipe,
number of selected objects: 1] (from bound object <NSTextField:
0x3357e0>): I just felt like doing something bad
```

More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Related Sessions

Cocoa and Core Data Feedback Forum	Session 139 Friday 9:00am
Advanced Core Data Usages	**Session 143** **Friday 2:00pm**
Writing Cocoa Automator Actions	Session 137 Thursday 3:30pm

Who to Contact

Matthew Formica
Developer Tools and Cocoa Evangelist
mformica@apple.com

Cocoa Dev
cocoa-dev@lists.apple.com

Q&A

Matthew Formica
Developer Tools and Cocoa Evangelist
mformica@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference